CBS1904 – Capstone Project

3D modeling and VR for enhanced teaching/learning and scientific simulation

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

COMPUTER SCIENCE WITH SPECIALIZATION IN BUSINESS SYSTEMS

by

21BBS0178 DEVANSH SAXENA

Under the Supervision of

Dr. KEYUR BHANUPRASAD JOSHI

ASSOCIATE PROFESSOR Sr.

School of Computer Science and Engineering

(SCOPE)

April 2025

DECLARATION

I hereby declare that the project entitled **3D modeling and VR for enhanced teaching/learning and scientific simulation** submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of Bonafide work carried out by me under the supervision of **Dr. Keyur Bhanuprasad Joshi.**

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree ordiploma in this institute or any other institute or university.

Place : Vellore

Date : 23/04/2025

Signature of the Candidate

Devansh Saxena

CERTIFICATE

This is to certify that the project entitled **3D modeling and VR for enhanced teaching/learning and scientific simulation** submitted by **Devansh Saxena(21BBS0178) School of Computer Science and Engineering**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of Bonafide work carried out by him / her under my supervision during Winter Semester 2024-2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute orany other institute or university. The project fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 23/04/'25

VELLORE INSTITUTE OF TECHNOLOGY (VIT) (Deemed to be University under section 3 of UGC Act, 1956) School of Computer Science & Engineering (SCOPE) Vellore-632014, T.N., India

Signature of the Guide

23/04/25

Internal Examiner

External Examiner

Dr. RAJKUMAR S

Computer Science and Engineering

EXECUTIVE SUMMARY

The integration **of 3D modeling and Virtual Reality (VR)** into educational and scientific domains represents a transformative advancement in how knowledge is delivered, experienced, and applied. These technologies offer immersive, interactive, and highly visual environments that significantly enhance learning outcomes, student engagement, and scientific exploration.

In education, VR and 3D modeling facilitate experiential learning by allowing students to visualize complex concepts, conduct virtual experiments, and explore environments otherwise inaccessible—such as the inside of a cell, historical landmarks, or distant planets. This approach supports diverse learning styles and boosts retention, comprehension, and motivation.

In scientific simulation, 3D models and VR environments enable researchers to simulate physical systems, visualize abstract data, and test hypotheses in risk-free, cost-effective settings. Applications span across fields such as medicine, engineering, physics, and environmental science, where virtual prototyping, anatomical modeling, and dynamic simulations provide critical insights and drive innovation.

Key features of the system include:

• **Immersive Visualization** Enables users to explore complex concepts and environments in a fully interactive 3D space for deeper understanding.
• **Real-Time Simulation** Supports dynamic, real-time modeling of scientific phenomena for experimentation and hypothesis testing.
• **Interactive Learning Modules** Offers customizable, scenario-based learning experiences tailored to various educational levels and subjects**.**
• **Collaborative Virtual Environments** Facilitates remote teamwork and shared exploration through multi-user VR spaces.

Adoption of these technologies fosters collaboration, scalability, and adaptability, preparing both educators and students for a digitally-driven future. However, successful implementation requires thoughtful integration into curricula, investment in hardware and software, and training for educators and researchers.

By leveraging the power of 3D modeling and VR, institutions can create more engaging, inclusive, and effective learning and research environments that bridge theory with real-world application.

ACKNOWLEDGEMENTS

I am deeply grateful to the management of Vellore Institute of Technology (VIT) for providing me with the opportunity and resources to undertake this project. Their commitment to fostering a conducive learning environment has been instrumental in my academic journey. The support and infrastructure provided by VIT have enabled me to explore and develop my ideas to their fullest potential.

My sincere thanks to Dr. Jaisankar N, Dean - School of Computer Science and Engineering (SCOPE), for his unwavering support and encouragement. His leadership and vision have greatly inspired me to strive for excellence.

I express my profound appreciation to Dr. RAJKUMAR S, the Head of the Computer Science and Engineering and Business Systems, for his/her insightful guidance and continuous support. His/her expertise and advice have been crucial in shaping throughout the course. His/her constructive feedback and encouragement have been invaluable in overcoming challenges and achieving goals.

I am immensely thankful to my project supervisor, **Dr. KEYUR BHANUPRASAD JOSHI**, for his dedicated mentorship and invaluable feedback. His patience, knowledge, and encouragement have been pivotal in the successful completion of this project. My supervisor's willingness to share his/her expertise and provide thoughtful guidance has been instrumental in refining my ideas and methodologies. His support has not only contributed to the success of this project but has also enriched my overall academic experience.

Thank you all for your contributions and support.

Devansh Saxena

ABSTRACT

The convergence of 3D modeling and Virtual Reality (VR) technologies has opened new frontiers in education and scientific research, offering immersive and interactive platforms that transform traditional learning and simulation practices. By enabling users to visualize, manipulate, and engage with complex data and environments in real-time, these technologies bridge the gap between theoretical knowledge and practical experience. In educational settings, they enhance conceptual understanding, foster engagement, and support differentiated learning strategies. In scientific domains, VR and 3D modeling provide powerful tools for simulating physical systems, testing hypotheses, and visualizing abstract phenomena with high fidelity.

This paper explores the key features, applications, and benefits of integrating these technologies into curricula and research workflows, highlighting their potential to drive innovation and improve outcomes across disciplines. Challenges related to implementation, accessibility, and scalability are also discussed, along with future directions for development and integration.

TABLE OF CONTENTS

List of Tables

Table No.	Title	Page No.

List of Figures

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CG	Computer Graphics
CPU	Central Processing Unit
DX	Digital Experience
GIS	Geographic Information System
GPU	Graphics Processing Unit
HMD	Head-Mounted Display
IoT	Internet of Things
LOD	Level of Detail
ML	Machine Learning
MR	Mixed Reality
SDK	Software Development Kit
UI	User Interface
UX	User Experience
VR	Virtual Reality
XR	Extended Reality

Symbols and Notations

Δ (*Delta*)	Represents change r dfference (Δt for change in time)
∇ (Nabla)	Gradient operator in vector calculus, often used in physics and simulations
δ (Partial Derivative)	Used to denote partial derivatives in simulation equations
	Indicating vector direction, $\bar{v}$ for velocity vector
$\|\|v\|$	Magnitude or norm of vector v
θ (Theta)	Commonly used for angles in rotation or orientation calculations
π (Pi)	Mathematical constant (~3.1416), used in geometry and trigonometry
ω (Omega)	Angular velocity in′simílematoi) used in stistical models
Σ (Sigma)	Summation symbol, frequently used in mathematical modeis
μ (*Mu*)	Coefficient of friction or mean in stat matcíx
t	Time variable in dynamic simulations
x, y, z	Rotation coordinates in 3D space
R	Rotation matrix or transformation matrix
T	Translation vector or matrix
$f(x)$	A function of x, often represent transformatior equations
α (Alpha)	Aften used for transparency or blending in graphics
λ (Lambda)	λ Lambda for eigenvaues or in lighting/shading models

1. INTRODUCTION

1.1 Background

The integration of 3D modeling and Virtual Reality (VR) into educational and scientific contexts is the result of decades of technological evolution in computing, graphics, and interactive media. Initially developed for industries such as engineering, design, and entertainment, 3D modeling has grown into a powerful tool for visualizing and manipulating digital representations of real-world and abstract objects. Similarly, VR originated in high-stakes training simulations, such as military flight training, but has become more accessible and user-friendly with the advent of consumer-grade headsets and affordable development platforms. These advancements have laid the groundwork for their current use in teaching and scientific simulation.

In educational settings, the ability to transform abstract concepts into immersive and interactive experiences has opened new pedagogical possibilities. Complex systems in biology, chemistry, physics, and history can be explored in three dimensions, offering students a deeper and more intuitive understanding than traditional textbooks or lectures can provide. VR allows learners to engage with environments they would not otherwise experience—such as the inner workings of a human heart, the vastness of space, or the reconstruction of ancient civilizations—thus supporting experiential learning and increasing motivation. Moreover, these tools cater to diverse learning styles, fostering inclusivity and engagement across a broad range of learners.

Scientific research has also been significantly enhanced by the adoption of 3D modeling and VR. These technologies allow researchers to simulate real-world systems with high fidelity, manipulate virtual prototypes, and visualize data in dynamic and meaningful ways. In disciplines like medicine, engineering, environmental science, and physics, researchers can conduct experiments, test hypotheses, and demonstrate phenomena within a safe, controlled, and cost-effective environment. VR-based simulations also facilitate collaborative research across geographical boundaries, enabling teams to work together in shared virtual spaces in real time.

Educational institutions and research organizations are increasingly integrating these technologies into their infrastructure, driven by a growing emphasis on digital literacy and innovation. Universities are establishing VR labs and digital fabrication spaces, while educational content creators are developing curriculum-aligned immersive modules for various subjects. However, successful implementation is not without its challenges. Educators must be trained in using and designing content for these platforms, and there must be investments in appropriate hardware and software. Accessibility and inclusivity also remain critical considerations, ensuring that these technologies benefit all learners regardless of background or ability.

Looking to the future, the convergence of VR and 3D modeling with other emerging technologies such as artificial intelligence, cloud computing, and haptic feedback is poised to further revolutionize teaching and scientific research. As development costs decrease and ease of use improves, these tools will become increasingly integral to both classroom learning and advanced simulation.

1.2 Motivations

The rapid advancement of digital technologies has exposed significant gaps in traditional teaching methods, particularly in conveying complex, abstract, or highly visual content. In fields such as science, engineering, and medicine, students often struggle to fully grasp spatial relationships, dynamic processes, or microscopic structures using static images or textual explanations alone. This project is motivated by the need to bridge that gap—**leveraging immersive 3D modeling** and VR to make learning more tangible, intuitive, and engaging for learners of all levels.

At the same time, the limitations of physical resources—such as access to laboratories, expensive materials, or specialized equipment—can constrain both teaching and research. VR offers a powerful solution by simulating these environments with high fidelity and minimal cost. It enables users to conduct experiments, explore models, and interact with data in ways that would otherwise be inaccessible, dangerous, or impractical. This not only democratizes access to quality learning experiences but also supports distance and inclusive education initiatives.

From a research perspective, the motivation also stems from the growing demand for interactive tools that can enhance the way scientific data is interpreted and hypotheses are tested. In simulation-heavy disciplines, the ability to visualize and interact with complex systems in 3D space accelerates innovation and deepens insight. By embedding real-time physics and data-driven models within immersive environments, researchers can observe outcomes, manipulate variables, and collaborate more effectively, even remotely.

Ultimately, this project aims to transform educational and scientific experience by making it more interactive, immersive, and accessible. The motivation lies in fostering deeper understanding, increasing student engagement, and enhancing research capabilities through the intelligent application of emerging technologies. By aligning 3D modeling and VR with curricular goals and scientific needs, the project aspires to build a future-ready learning and research ecosystem that prepares individuals not just to absorb information, but to explore, create, and innovate within it.

1.3 Scope of the Project

The aims to use 3D modelling and virtualization to enhance the teaching process and make the whole process more easy to understand while also focusing that a sense of interaction is also present. The project's scope is extensive and focuses on several critical areas:

1. **Development of 3D Educational Models-** Design and create interactive 3D models of scientific and academic concepts (e.g., anatomical structures, chemical compounds, physical systems) for use in classroom and research settings.

2. **Integration of VR Learning Environments**– Build immersive Virtual Reality environments that simulate laboratories, natural phenomena, and historical or conceptual spaces to enhance experiential learning.

3. **Cross-Disciplinary Application** – Apply the technology across multiple disciplines such as biology, physics, chemistry, engineering, medicine, and environmental science.

4. **User-Centered Interface Design** – Develop intuitive and accessible user interfaces for both students and instructors, ensuring usability for diverse learning levels and backgrounds.

5. **Interactive Simulation and Experimentation** – Enable real-time interaction with models and simulations to allow learners and researchers to manipulate variables and observe dynamic outcomes.

6. **Support for Remote and Inclusive Learning** – Facilitate distance education by enabling full VR experiences and simulations accessible via the internet, supporting students in remote or under-resourced locations.

7. **Scalable Platform for Content Expansion** – Design the system architecture to support future content creation, integration of AI features, and collaborative virtual environments for group learning and research.

2. PROJECT DESCRIPTION AND GOALS

2.1 Literature Review

The integration of 3D modeling and Virtual Reality (VR) into education and scientific research has significantly evolved over the past decade, reshaping the way knowledge is transmitted, engaged with, and applied. These technologies are increasingly recognized for their potential to enhance learning and foster deeper understanding, particularly in disciplines that involve complex, spatial, or abstract concepts. 3D modeling, which began in fields like engineering and architecture, has expanded into education by allowing students to visualize and interact with digital representations of real-world and theoretical objects. By offering visualizations of concepts like molecular structures or anatomical systems, 3D modeling provides students with an immersive, hands-on experience, facilitating better retention and understanding. This type of learning caters to diverse educational needs, enhancing engagement by offering dynamic, interactive models that go beyond the limitations of traditional textbooks or static diagrams (Azuma, 2019).

Virtual Reality, in contrast, introduces an even greater level of immersion, enabling students to engage in simulated environments that replicate real-world experiences or present entirely new, conceptual spaces. VR's immersive nature allows learners to explore environments, conduct virtual experiments, and interact with data in ways that were previously unattainable. Studies have demonstrated that VR can enhance students' motivation, conceptual understanding, and overall engagement. For example, Mikropoulos and Natsis (2011) highlighted the effectiveness of VR in subjects like anatomy and physics, where it allowed students to visualize and manipulate complex structures in 3D. Moreover, VR provides an experiential learning opportunity that caters to different learning styles, fostering an inclusive environment for diverse learners and boosting overall comprehension and retention of knowledge.

In the context of scientific research, 3D modeling and VR serve as powerful tools for simulating complex systems and conducting experiments that are either impractical or too costly in real-life scenarios. For example, in fields like medicine, VR has revolutionized surgical training by offering a risk-free environment where medical professionals can practice procedures and refine their skills (Rosen et al., 2019). Similarly, in engineering and environmental sciences, 3D models allow researchers to visualize prototypes, test their viability, and simulate various conditions without the need for physical experimentation. Such simulations are not only more cost-effective but also allow for the manipulation of variables in ways that traditional experiments cannot, facilitating deeper insights and accelerated innovation. This ability to simulate dynamic systems is particularly valuable in research fields such as chemistry, physics, and environmental science, where phenomena can be complex, abstract, or difficult to visualize.

However, while the benefits of these technologies are clear, there are several challenges to their widespread adoption, particularly in educational institutions and research labs. The cost of VR hardware and the necessary computational power remain a significant barrier, especially for institutions with limited budgets. Freeman et al. (2017) pointed out that, despite the decreasing prices of VR systems, the financial investment required for large-scale adoption of these technologies is still a considerable obstacle. Moreover, the development of high-quality content

that aligns with curricula remains a challenge. Many existing VR applications are not yet fully integrated into educational frameworks, and creating new, curriculum-specific VR and 3D content requires specialized knowledge in both software development and subject matter expertise. This barrier can limit the effective implementation of these technologies in classrooms and labs.

Despite these challenges, ongoing research and technological advancements suggest a bright future for the use of 3D modeling and VR in education and scientific research. As the technology continues to evolve, emerging trends such as the integration of Artificial Intelligence (AI) and machine learning with VR hold significant potential. These innovations could lead to more personalized, adaptive learning environments that cater to the individual needs of students, further enhancing engagement and outcomes. Additionally, the integration of haptic feedback, which allows users to physically "feel" virtual objects, is poised to make VR environments even more immersive and interactive. These advancements are expected to overcome many of the current barriers, making VR and 3D modeling more accessible and effective in a variety of disciplines.

Looking ahead, the convergence of VR with other emerging technologies, such as cloud computing and real-time data analytics, will only further enhance its potential. As VR systems become more affordable and user-friendly, the technology will likely become a staple in classrooms and research institutions worldwide. By providing opportunities for interactive, hands-on learning and real-time scientific simulations, VR and 3D modeling are poised to revolutionize both education and research. The ongoing development and refinement of these technologies promise to unlock new opportunities for collaboration, remote learning, and scientific discovery, shaping a future where immersive, interactive environments are central to how we learn and explore the world around us.

2.2 Research Gap

Despite advancements in modelling softwares and immersiveness of VR headsets, several challenges and limitations persist. Below are three key research gaps that require further exploration to improve the efficiency and practicality of Virtual Air Writing Systems.

1. Lack of Pedagogical Frameworks for VR Integration

While VR and 3D modeling have shown great promise in enhancing learning outcomes, one of the significant research gaps is the absence of robust pedagogical frameworks that guide their effective integration into curriculum design. Most VR implementations are experimental or supplementary, with limited alignment to established teaching methodologies such as constructivism, inquiry-based learning, or Bloom’s Taxonomy. Educators often lack a clear understanding of how to design lessons that effectively incorporate VR to support learning objectives. This leads to a mismatch between the potential of the technology and its actual classroom impact. Research is needed to develop evidence-based frameworks that guide teachers in choosing the right VR content, sequencing learning experiences, and assessing student performance in virtual environments.

2. Insufficient Longitudinal Studies on Learning Outcomes

Another critical research gap is the scarcity of longitudinal studies that evaluate the long-term impact of VR and 3D modeling on learning outcomes and skill development. Most existing studies focus on short-term engagement, immediate comprehension, or novelty effects, without tracking how these technologies affect knowledge retention, critical thinking, or real-world application over time. As a result, it remains unclear whether the benefits observed in short experiments persist in the long term or translate into practical competencies. Research that spans multiple semesters or academic years, and involves control groups, is essential to understand the sustained educational value and the return on investment of adopting immersive technologies in education and scientific training.

3. Accessibility and Equity Issues in VR Implementation

A significant gap also exists in addressing accessibility and equity in the deployment of VR and 3D modeling tools across diverse educational settings. Many studies focus on well-funded institutions or pilot programs, leaving a void in understanding how these technologies can be effectively and equitably implemented in under-resourced schools or remote areas. Questions remain about how to ensure inclusivity for students with disabilities, those lacking access to high-end devices, or those from socioeconomically disadvantaged backgrounds. Moreover, most VR content is developed in dominant global languages and cultural contexts, limiting its relevance and effectiveness in diverse educational environments. Future research should explore affordable, adaptable, and inclusive VR solutions that bridge the digital divide and ensure equitable access to immersive learning experiences.

2.3 Objectives

1. **Enhance Conceptual Understanding:** To provide students with an interactive and visual learning experience that helps them better understand complex physics concepts such as motion, force, energy, and electromagnetism, which are often abstract in traditional teaching.

2. **Bridge the Gap Between Theory and Application:** To simulate real-world physics phenomena in a controlled digital environment, helping students make the connection between mathematical formulas and their practical implications or outcomes.

3. **Promote Independent and Collaborative Learning:** To support both individual learning and group collaboration by including multi-user options, shared scenarios, or discussion prompts within the simulation environment.

2.4 Problem Statement

For many students, including Arun—a high school learner with a keen interest in technology—grasping basic physics concepts has proven to be a frustrating and demotivating experience. Despite his efforts to study from textbooks, attend lectures, and complete assignments, the abstract nature of core topics like motion, force, and energy continues to elude him. He often finds himself memorizing formulas without fully understanding their meaning or practical application, leading to poor performance in assessments and a growing sense of disconnection from the subject.

Arun's difficulties stem largely from the traditional methods of instruction that rely heavily on passive learning and theoretical explanations. In his classroom, physics is taught using static diagrams, complex mathematical derivations, and limited lab sessions, which provide the dynamic, real-world context that Arun needs to engage with the material. Without the opportunity to explore concepts visually or interactively, he struggles to build a mental model of how physical phenomena behave. This has not only weakened his academic foundation but also eroded his confidence and interest in pursuing STEM fields further.

The lack of accessible, experiential learning tools in his educational environment has made it challenging for students like Arun to bridge the gap between theory and understanding. This problem calls for a new approach—one that leverages interactive simulations and immersive technologies to transform physics from an abstract puzzle into a tangible, intuitive experience. By incorporating 3D modeling and virtual simulations, students would be empowered to experiment, visualize outcomes, and actively learn by doing—making concepts more relatable, retention stronger, and education more inclusive.

2.5 Project Plan

1. **Research and Planning**
   Goal: Understand the educational needs, identify technical requirements, and review existing solutions to inform a strategic development approach.
   Actions: Defining clear educational goals aligned with curriculum standards, and outlining the overall project scope, timeline, and required resources.
2. **System Design**
   Goal: Create a clear blueprint for both the user experience and system architecture of the simulator.
   Actions: Designing conceptual layouts and interactive flows for the simulator, mapping physics concepts to 3D visualization and interaction models, choosing appropriate frameworks for development and VR support.
3. **Development**
   Goal: Build the functional components of the simulator and implement the visual and interactive elements.
   Actions: Constructing the physics simulation engines and interactive components using chosen development tools, creating 3D assets and environments to represent physical systems, integrating variable controls and dynamic visual feedback for real-time exploration
4. **Integration and Testing**
   Goal: Ensure all system components work together smoothly and meet educational goals.
   Actions:Combining individual simulation modules into a cohesive application framework, testing user interactions and physics accuracy across scenarios, observing real users navigating the simulation to identify usability challenges, fixing bugs and optimizing responsiveness
5. **Deployment**
   Goal: Launch the simulator in a real or pilot educational environment for practical use.
   Actions: Preparing platform-specific deployment packages for PC, mobile, or VR devices, delivering user manuals or introductory tutorials to support usage, conducting

pilot sessions in classrooms or labs, collecting early feedback from both students and instructors.

6. **Evaluation and Iteration**
   Goal: Assess the educational effectiveness and continuously improve the system based on feedback and performance data.
   Actions: Measuring the impact of the simulator through assessments and surveys, analyzing user engagement and interaction logs to identify pain points, incorporating suggestions from educators into future updates, revising content and interface based on learning efficacy and usability

This plan ensures we develop a model that's not only technologically advanced but also intuitive and accessible for all users.

3. TECHNICAL SPECIFICATION

3.1 REQUIREMENTS

The technical implementation of the physics simulator requires a combination of high-performance computing, intuitive user interfaces, and immersive visualization tools to support interactive and educational experiences. Core requirements include a robust physics engine capable of accurately simulating Newtonian mechanics, force interactions, and motion dynamics in real time. The system should be built using a versatile 3D development platform such as Unity or Unreal Engine, with support for VR headsets and cross-platform deployment. To ensure smooth performance, the simulator will require hardware with sufficient GPU processing power and memory, especially when rendering complex 3D scenes. The interface must be user-friendly, offering drag-and-drop elements, sliders for variable manipulation, and real-time feedback displays for measurements like velocity, acceleration, and energy. Additionally, the simulator should support modular content management to allow expansion with new topics, along with cloud integration for remote access and data tracking. Accessibility features, such as language options and voice support, should also be considered to accommodate diverse users. The overall system must ensure scalability, maintainability, and educational alignment, providing a seamless and engaging learning tool for students across varying skill levels.

3.1.1 Functional Requirements

- **Variable Manipulation:** The system must allow users to manipulate physical variables such as mass, force, and velocity in real time to observe their effects on motion.
- **Accurate Physical Simulation:** The simulator must accurately visualize the outcomes of physical interactions based on classical mechanics principles, including gravity, friction, and collisions.
- **User-Friendly Interface:** The interface must support intuitive input controls such as sliders, buttons, and object dragging for ease of use by students with varying technical skills.
- **Real-Time Feedback:** The system must provide real-time feedback through visual indicators and numerical data outputs for parameters like speed, acceleration, and kinetic energy.
- **Immersive VR Support:** The simulator must support immersive VR mode to allow students to interact with experiments in a 3D spatial environment for deeper conceptual understanding.
- **Modular Content System:** The system must include a module framework that enables instructors or developers to add, modify, or remove simulation scenarios based on curriculum needs.

3.1.2 Non-Functional Requirements

- **Performance Efficiency:** The simulator must maintain smooth performance and low latency, especially during real-time simulations and VR interactions.

- **Cross-Platform Compatibility:** The system should operate consistently across different operating systems and devices, including desktops, laptops, and VR headsets.
- **Scalability:** The architecture must support the addition of new simulation modules and features without requiring major changes to the core system.
- **Usability:** The application should be intuitive and accessible to users with minimal training, providing clear navigation and responsive feedback.
- **Reliability and Stability:** The simulator must operate reliably under normal usage conditions, with mechanisms in place to handle errors and prevent crashes during extended sessions.
- **Security and Data Privacy:** The system must ensure secure handling of user data, including protection of login credentials and compliance with data privacy standards where applicable.

3.2 Feasibility Study

A feasibility study helps determine whether the proposed Virtual Gesture Recognition System is practical and viable for implementation. The following factors have been evaluated:

3.2.1 Technical Feasibility

The technical feasibility of the physics simulator project is highly promising given current advancements in graphics processing, real-time physics engines, and VR hardware. Modern game development platforms like Unity and Unreal Engine offer powerful physics simulation capabilities and built-in support for VR environments. These platforms also include large libraries of reusable assets and tools that significantly reduce development time while ensuring high performance and graphical fidelity. Since these engines support scripting in widely known languages like C# and Python (for integrations), your development team can prototype and scale features efficiently.

The hardware requirements for this project, such as VR headsets (e.g., Meta Quest, HTC Vive) and mid- to high-range computers, are now commercially accessible and widely adopted in educational institutions. Most institutions already have or can feasibly acquire the minimum hardware required to run simulations smoothly, especially with modular optimization techniques like Level of Detail (LOD) and GPU-based rendering. For broader accessibility, the simulator can also be built with scalable rendering options to support both 2D and 3D modes depending on user hardware.

From a software integration perspective, the project is feasible due to the availability of open-source or affordable tools for 3D modeling (like Blender), physics libraries (e.g., Bullet, NVIDIA PhysX), and educational content management systems. These tools allow seamless development and deployment of educational content that aligns with curricular goals. Furthermore, the integration of data logging, UI customization, and performance tracking can be managed through modular APIs and SDKs, ensuring smooth extensibility.

Another aspect of feasibility lies in the development team's technical capacity. With access to interdisciplinary collaboration—including educators, developers, and designers—this project is well-positioned for success. Most programming tasks, such as real-time variable control, collision

detection, and user interaction logic, are well-documented and widely implemented in existing development communities, allowing for support and knowledge sharing during development.

In conclusion, the technical environment required to develop and deploy the simulator is readily available, well-supported, and aligns with current trends in educational and immersive technology. With careful project planning and resource allocation, the simulator can be built to deliver engaging, scientifically accurate, and user-friendly learning experiences for students across various educational levels.

3.2.2 Economic Feasibility

The economic feasibility of the physics simulator project is promising, with a clear understanding of both development and operational costs. Initial expenses will include software development, 3D modeling, VR content creation, and hardware acquisition (VR headsets and compatible computers). These costs may range from **Rs. 35,00,000 to Rs.1,50,00,00,000** depending on the scope of the project. Additional costs will come from testing, cloud infrastructure for remote access, and ongoing maintenance, which could amount to **Rs.15,00,000 to Rs.30,00,000** annually. However, using open-source tools and cloud-based services could help reduce the overall financial burden.

From a revenue perspective, the project has strong potential through subscription-based licensing models, grants for educational innovation, and possible SaaS offerings for institutions. Educational institutions can benefit from a cost-effective, scalable solution that enhances learning outcomes while reducing the need for expensive physical labs. Over time, **the long-term benefits**, including improved student engagement and the ability to update content regularly, will provide a strong return on investment, making **the project financially viable and sustainable**.
Potential for commercial applications in gaming, healthcare, education, and smart homes can lead to high return on investment (ROI). The project is therefore considered **economically feasible**.

3.2.3 Social Feasibility

The social feasibility of the physics simulator project is highly favorable, particularly in its capacity to improve science education and promote equal access to quality learning experiences. One of the primary social benefits lies in its ability to engage students who struggle with traditional learning methods. By using interactive simulations and VR environments, the project caters to visual, kinesthetic, and experiential learners, helping bridge the gap for students who may be left behind in conventional classroom settings. This can lead to improved understanding, greater academic confidence, and ultimately, better performance in physics and related subjects.

Moreover, the platform promotes inclusivity and accessibility across different geographic and socioeconomic backgrounds. Students in remote or underfunded schools who lack access to fully equipped laboratories can experience the same high-quality educational simulations as those in more privileged institutions. By offering a virtual, immersive alternative to hands-on experiments, the system reduces reliance on physical infrastructure, making advanced learning tools available to a broader audience. This can play a significant role in reducing educational inequality and fostering a more level playing field.

Social acceptance and enthusiasm for the technology are also likely to be strong, given the increasing integration of digital tools into everyday life and education. As students and teachers become more comfortable with technology, the physics simulator aligns well with current educational trends toward digital transformation and active learning. It encourages collaboration, creativity, and curiosity—key values in 21st-century education—making it a socially impactful project that can transform not just how physics is taught, but how it is understood and appreciated by learners around the world.

3.3 System Specifications

3.3.1 Hardware Requirements

Client-Side Hardware Requirement

1. Processor (CPU):A minimum of Intel Core i5 (10th Gen or higher) or AMD Ryzen 5 3600 is recommended to handle the simulation computations smoothly. For optimal performance, higher-end CPUs like the Intel Core i7/i9 or AMD Ryzen 7/9 are ideal.
2. Graphics Card (GPU): For 3D rendering and VR support, a NVIDIA GTX 1660 or RTX 2060 (or equivalent AMD GPU like RX 5600 XT) is required at minimum. For VR-heavy applications, a NVIDIA RTX 3060 or higher is recommended.
3. RAM (Memory): At least 8 GB of RAM is required, with 16 GB or more recommended for optimal multitasking and rendering large simulations.
4. Storage:256 GB SSD minimum for quick loading and responsiveness. A 512 GB or higher SSD is preferable, especially if multiple simulation modules are used.
5. Display and VR Headset (Optional):A Full HD monitor (1920x1080) for desktop usage. For VR interaction, compatible headsets like Meta Quest 2, HTC Vive, or Valve Index should be used, along with sufficient USB and DisplayPort/HDMI outputs.

Server-Side Hardware Requirement

1. Processor (CPU): A multi-core server-grade processor such as Intel Xeon Silver/Gold or AMD EPYC is recommended to handle simultaneous user requests, simulation logic.
2. Memory (RAM): At least 32 GB DDR4 ECC RAM is required for smooth operation, with 64 GB or more recommended for larger simulations or concurrent user access.
3. Graphics (GPU): If the server is involved in rendering or streaming 3D content (e.g., for thin clients or cloud rendering), a NVIDIA RTX A4000 or A6000, or Tesla-series GPU is essential. For simple backend processing without rendering, GPU may not be critical.|
4. Storage: A 1 TB NVMe SSD is recommended for fast access to assets, models, and databases. Additional HDDs or cloud storage can be used for long-term archival.

3.3.2 Software Requirements

To ensure compatibility and ease of execution, the system must use:

Web Browser: Latest versions of Chrome, Firefox, or Edge with support for WebGL and WebXR (for VR compatibility).
VR Runtime (if applicable): SteamVR, Oculus Runtime, or Windows Mixed Reality based on the headset in use.
Operating System: Windows 10/11, macOS (with Metal support), or Linux (Ubuntu 20.04+) depending on the user's device.
3D Plugin Support: Optional installation of browser plugins or drivers that enhance 3D/VR capabilities if native support is limited.

3.4 Constraints

The development of a 3D modeling and VR-based physics simulation system comes with several notable constraints that need to be considered throughout the project lifecycle. One of the primary limitations is hardware dependency, as users require VR headsets or high-performance systems with proper GPU support to fully experience immersive simulations, which may not be accessible to all students. Additionally, browser compatibility and support for WebGL/WebXR features vary, potentially affecting the consistency of user experience across platforms. The project also faces resource constraints in terms of development time, availability of skilled personnel for 3D modeling and VR integration, and the potential learning curve associated with advanced tools like Unity or Unreal Engine. Furthermore, internet bandwidth and latency could affect performance, especially during real-time simulations or when accessing server-hosted content. Lastly, content accuracy and educational effectiveness must be rigorously ensured to meet pedagogical goals, posing a challenge in balancing interactivity with correctness and clarity in the simulation models.

3.5 Assumptions and Dependencies

In planning and developing a 3D modeling and VR-based physics simulation system, several assumptions and dependencies underpin the project's success and feasibility.

Assumptions: One core assumption is that users (students and educators) will have access to compatible hardware, such as VR headsets or at least computers capable of handling 3D graphics smoothly. It's also assumed that internet connectivity will be stable and fast enough to support real-time simulations and data exchange, especially for remote or web-based experiences. The project assumes willingness and ability of educators to integrate the platform into their curriculum, including a baseline level of digital literacy. Furthermore, it's presumed that 3D simulations will effectively improve conceptual understanding of basic physics, making the investment in technology pedagogically justified. Another assumption is that content and models developed will be reusable and scalable, enabling future expansion.

Dependencies: This project is dependent on several technical and operational factors. Technologically, it depends on third-party software frameworks and libraries such as Unity, Unreal Engine, or WebXR APIs, which must remain stable and well-supported during the development and maintenance phases. It also relies on regular updates and compatibility with operating systems and browsers, particularly for web-delivered components. On the institutional side, the project depends on collaboration between domain experts and developers

to ensure accurate simulation of physics principles. Additionally, server-side infrastructure and cloud services are critical for hosting, rendering, and distributing VR content, implying reliance on external hosting providers or institutional IT services. Lastly, the system's success hinges on user feedback and iterative improvement, meaning ongoing engagement from students and faculty is essential to refine and adapt the system post-deployment.

4. DESIGN APPROACH AND DETAILS

This section focuses on the **architectural design, project timeline, and workflow processes** necessary for implementing **in the Modelling and Virtualization**. It includes the **Gantt Chart, Work Breakdown Structure (WBS), and System Architecture Diagram** to illustrate project execution and model deployment strategies.

4.1 SYSTEM ARCHIETECTURE [Gantt Chart and Work Breakdown Structure (WBS)]

The **Gantt Chart** represents the **project timeline**, ensuring that key tasks such as **Software familiarization, creating models, developing schenarios, and performance evaluation** are completed on schedule. The **Work Breakdown Structure (WBS)** provides a structured **task hierarchy**, categorizing project components into different execution phases.

The **major phases of project execution** for the **Virtual Air Writing System** include:

1. Requirement Analysis and Planning

- Identify student and educator needs
- Analyze existing physics curriculum
- Define scope and key deliverables
- Conduct initial feasibility assessment

2. System Design and Architecture

- Design user interface and interaction flow
- Define 3D environment structure
- Choose suitable physics engine and VR framework
- Plan integration architecture (client-server model)

3. 3D Content Development and VR Environment Setup

- Model key physics concepts in 3D (e.g., motion, gravity)
- Design immersive learning scenarios
- Optimize models for VR performance
- Import assets into VR development environment

4. Implementation and Coding

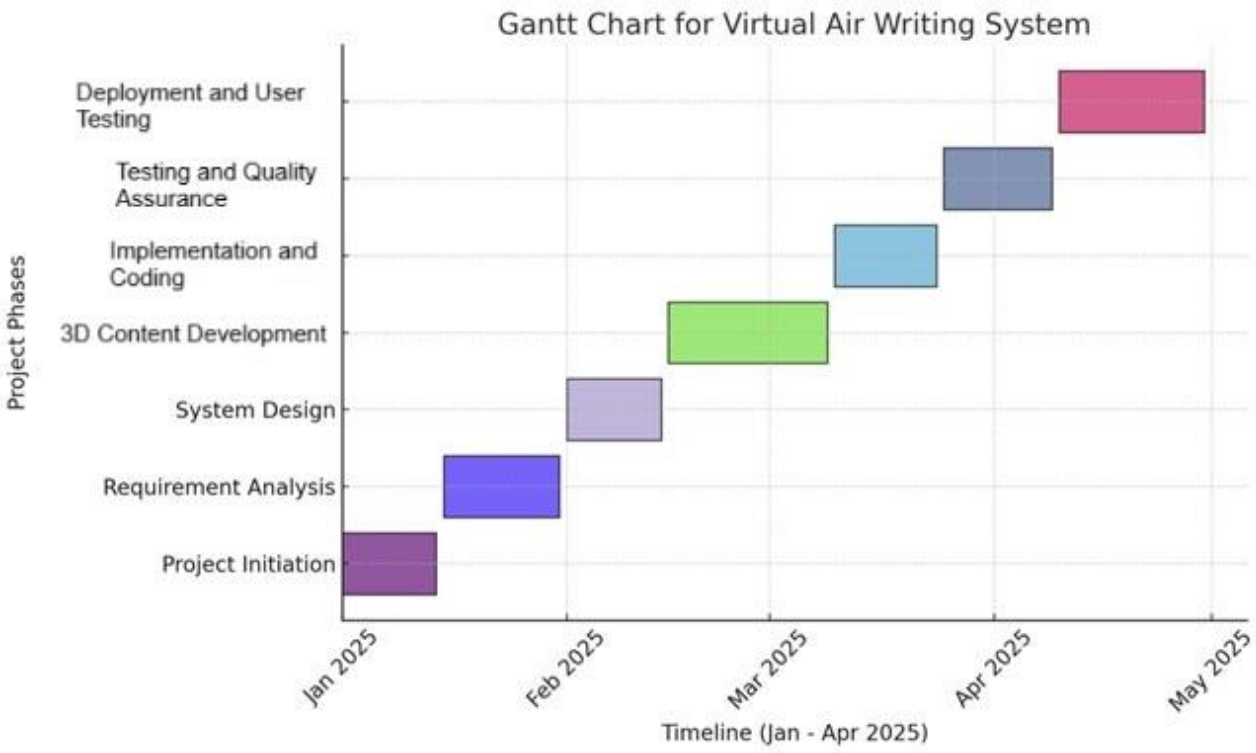
- Code simulation behavior and logic
- Integrate physics calculations and visual feedback
- Develop interactivity features (drag, drop, manipulate)
- Set up real-time feedback and result visualization

5. Testing and Quality Assurance

- Conduct unit and integration testing
- Perform cross-device compatibility checks
- Get feedback from student test users
- Debug and fix identified issues

6. Deployment and User Training

- Set up deployment server or distribution platform
- Launch pilot version in educational setting
- Create user manuals and tutorials
- Train instructors and gather feedback



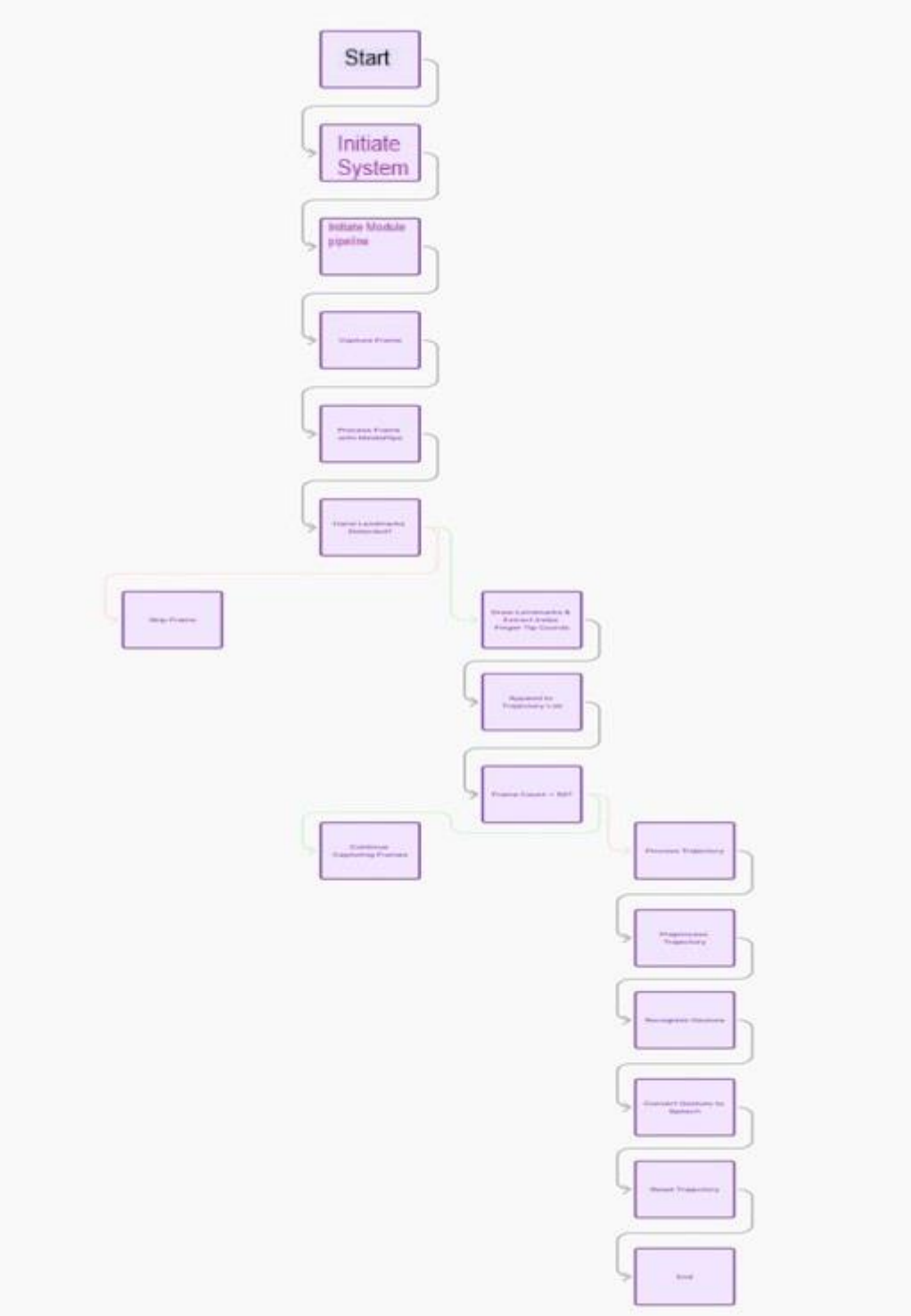
4.2 DESIGN

4.2.1 Data flow diagram

Workflow diagram for the **Virtual Air Writing System**:

* **User Input** → The user writes in the air using a motion-sensing device.
* **Motion Tracking** → Sensors track hand movements and gestures.
* **Data Processing** → The system converts motion data into digital signals.

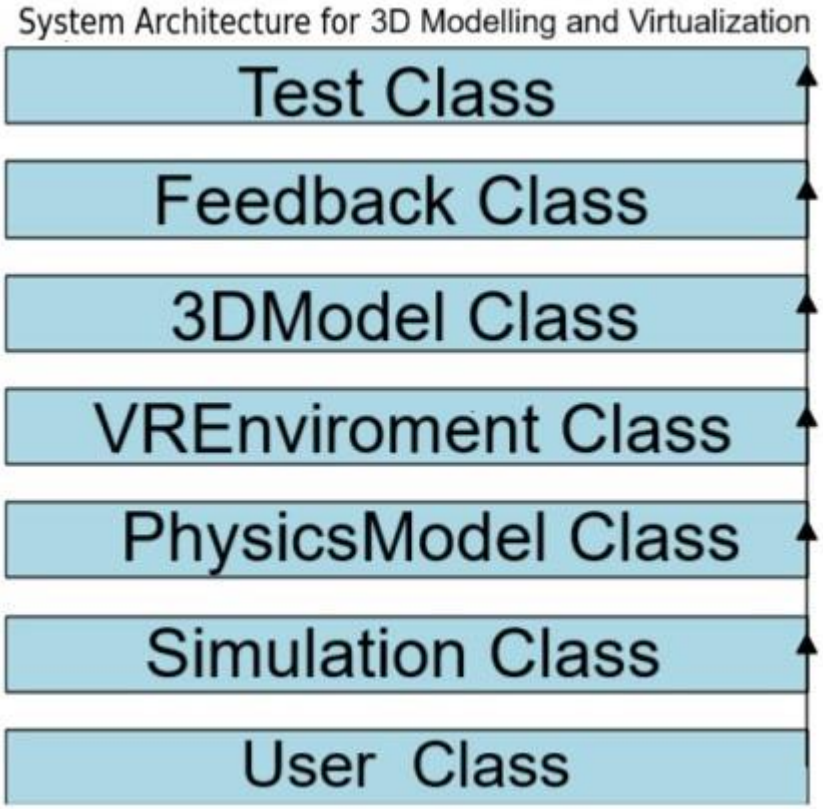
* **Character Recognition** → AI algorithms identify letters and words. * **Text Output** → The recognized text is displayed on a digital screen. * **Editing & Saving** → The user can edit and save the text if needed.



4.2.2 ***Class Diagram***

The virtual air writing class diagram comprises:

- **User Class:** Represents the users interacting with the VR simulation (students and educators).
- **Simulation Class:** Responsible for managing the physics simulations and handling user inputs.
- **PhysicsModel Class:** Represents different physics models used in the simulation (e.g., gravity, motion).
- **VREnvironment Class:** Manages the creation and rendering of the virtual reality environment for immersive experiences.
- **3DModel Class:** Represents the 3D models used in the VR simulation (e.g., objects, characters, scientific structures).
- **Feedback Class:** Manages the collection of user feedback and simulation results.
- **Test Class:** Handles the testing and debugging of the system, including performance and user experience testing.



5.METHODOLOGY AND TESTING

5.1 Module Description

The methodology for developing the 3D physics VR simulation involves a structured approach that integrates both theoretical and practical aspects of physics into an immersive virtual environment. The process begins with the design and creation of accurate 3D models representing physical phenomena and environments. These models are integrated into a VR environment that allows users to interact with them in real-time. The system uses a physics engine to simulate real-world interactions, applying laws of motion, gravity, and other physical principles. The goal is to create an engaging and accurate learning tool that improves the understanding of fundamental physics concepts by providing a hands-on, immersive experience. The development follows an iterative process, beginning with prototyping, followed by user feedback collection, testing, and refinement to ensure the simulation is accurate, intuitive, and effective for educational purposes.

Testing for this VR-based simulation is crucial to ensure both its functionality and educational effectiveness. A combination of unit testing, integration testing, and user acceptance testing (UAT) will be employed to evaluate the software's performance and reliability. Unit testing ensures individual components like physics models, user interaction mechanisms, and VR rendering functions work correctly. Integration testing checks how these components function together in the overall system. UAT focuses on gathering feedback from students and educators to evaluate user experience, usability, and the educational impact of the simulation. During the testing phase, special attention will be given to ensuring a smooth and immersive user experience, without motion sickness or technical glitches, and to verify that the physics simulations are both accurate and educational. Additionally, performance testing will evaluate the system's response under different load conditions, ensuring the platform can handle varying numbers of users simultaneously.

5.1.1 User Interface Module

Purpose:

- To provide an intuitive interface through which users (students and educators) can interact with the simulation.
- To simplify navigation through various educational tools, ensuring that the user experience is streamlined and user-friendly.
- To ensure the interface is customizable, allowing educators to adapt the environment and tools for specific learning outcomes or teaching preferences.

Theory:
This module focuses on the design of a user-friendly interface that simplifies user interaction with the VR system. The core theory behind this module is based on Human-Computer Interaction (HCI), ensuring that the virtual environment is easy to navigate and provides clear visual cues. It also considers the concept of affordance, which ensures that interface elements suggest their functionality naturally.

5.1.2 Physics Simulation Engine

Purpose:

- To simulate physical interactions and behaviors within the virtual environment, ensuring accurate real-world behavior.
- To provide an adaptable system capable of simulating different physics concepts such as motion, energy, forces, and collisions.
- To allow real-time adjustments and testing of variables so that users can experiment with different physics parameters and directly observe outcomes.

Theory: The theory behind this module is grounded in classical mechanics, particularly Newtonian physics. This module will use principles such as force, motion, gravity, and energy conservation to simulate realistic physical interactions. The physics engine will be responsible for applying these principles dynamically, adjusting the simulation based on user input (e.g., adjusting the force applied to an object or the direction of motion).

5.1.3 3D Graphics and Environment Rendering

Purpose:

- To create and render 3D models of physical objects, environments, and interactive elements in the virtual reality space.
- To ensure high-quality visualizations, making the virtual world as realistic and immersive as possible to enhance learning.
- To dynamically adjust environmental details (lighting, textures, objects) based on user interaction, ensuring realistic responses to changes in the simulation.

Theory: This module is based on 3D computer graphics and rendering algorithms such as polygonal modeling, texture mapping, lighting, and shading. The theory of Ray Tracing or Rasterization will be applied to generate realistic visuals, ensuring that objects within the virtual space respond accurately to light and shadow, creating a believable immersive environment.

5.1.4 User Interaction and Control

Purpose:

- To allow users to interact with and manipulate objects within the VR simulation using input devices (e.g., VR controllers, hand tracking).
- To offer a range of customizable controls for different learning scenarios, including specialized interactions for specific physics topics.
- To track and record user actions for assessment and personalized feedback, enabling deeper learning through action-based learning models.

Theory: The core theory behind this module is interaction design and gesture recognition. It relies on affordance theory, which states that controls in the VR space should suggest their functionality. Additionally, kinesthetic learning theory is applied here, where users learn more effectively when they physically interact with objects, promoting deeper understanding of physics concepts through hands-on manipulation.

5.1.5 Feedback and Assessment Module

Purpose:

- To collect and analyze user performance data, provide feedback, and assess the user's understanding of the simulated physics concepts.
- To offer real-time evaluations of user progress, highlighting areas of improvement and reinforcing key concepts.
- To provide personalized feedback based on individual user interactions and outcomes, fostering a deeper understanding of concepts and improving retention.

Theory: The theoretical basis for this module is rooted in formative assessment and feedback theory. Immediate feedback helps reinforce learning by allowing users to understand mistakes and correct their misconceptions in real-time. Additionally, the theory of self-regulated learning suggests that feedback helps students take control of their learning process, encouraging reflection and improvement.

5.1.6 System Integration and Performance Optimization

Purpose:

- To ensure smooth integration of all system components and optimize performance, especially for VR applications.
- To optimize computational resources to ensure the system runs efficiently even with complex 3D models and simulations.
- To ensure that the system maintains high responsiveness, reducing latency, and improving the overall VR experience for users.

Theory: This module applies theories from system architecture and optimization techniques. The system will need to manage multiple simultaneous interactions and ensure that the simulation runs efficiently even with complex 3D environments. The use of multithreading and parallel computing for efficient processing and level-of-detail (LOD) algorithms will be crucial for balancing visual fidelity and system performance in real-time. Additionally, performance metrics such as frame rate stability and latency minimization will be prioritized to avoid VR motion sickness and ensure a smooth experience.

5.2 TESTING

The testing of the project will be a crucial phase to ensure its functionality, usability, and performance meet the required standards. It will involve several stages, including unit testing, integration testing, and system testing. Unit testing will focus on evaluating individual components such as the physics simulation engine, user interface, and 3D rendering system to ensure they work independently as expected. Integration testing will then verify the seamless interaction between these components, ensuring that they collaborate correctly in the virtual environment. System testing will assess the entire platform's performance, including user experience and response times, particularly in a VR context. Usability testing will be carried out to gauge the interface's intuitiveness and accessibility, with real users providing feedback on their experience.

Additionally, performance testing will evaluate the system's ability to handle complex simulations and maintain high responsiveness without significant latency or crashes. Throughout all these phases, continuous feedback will be incorporated to improve the system and address any issues before deployment.

5.2.1 Unit Testing

Unit testing will focus on verifying individual components of the system, such as the physics engine, user interface, and 3D rendering module. Each unit will be tested independently to ensure it functions correctly on its own, with attention given to edge cases and potential errors. This will ensure that no individual module has inherent bugs before they are integrated into the larger system.

5.2.2 Integration Testing

Integration testing will assess how well the individual modules work together. For instance, it will evaluate how the physics simulation interacts with the 3D environment and whether the user interface responds appropriately to user inputs. This stage ensures that the interactions between components are smooth and that data flows correctly between them.

5.2.3 System Testing

System testing will evaluate the overall functionality of the complete project. This includes assessing the entire system's ability to simulate physical phenomena accurately and provide an immersive experience. Testing will involve simulating various scenarios and checking if all elements of the system work in harmony, providing the expected outputs and user interactions.

5.2.4 Performance Testing

Performance testing will measure the system's efficiency and speed under various loads. It will assess whether the system can handle complex simulations without lagging or crashing. The system's performance under different VR conditions will be evaluated to ensure it can maintain smooth frame rates, especially when handling resource-intensive tasks like real-time physics simulations.

5.2.5 VR Environment Testing

Usability testing will gather user feedback on the interface and overall user experience. The goal is to ensure the system is intuitive and accessible, particularly for users who may not be familiar with VR or 3D modeling tools. Testers will assess how easy it is to navigate the interface, use tools, and interact with simulations, identifying any barriers to smooth user interaction.

5.2.6 VR Environment Testing

VR environment testing will specifically assess how the project performs in a Virtual Reality setting. This includes checking for user comfort, responsiveness, and motion sickness prevention. The project will be tested for visual fidelity, interaction latency, and the realism of the immersive experience. This testing ensures that users feel comfortable and engaged while interacting with the simulations in VR**.**

6. PROJECT IMPLEMENTATION

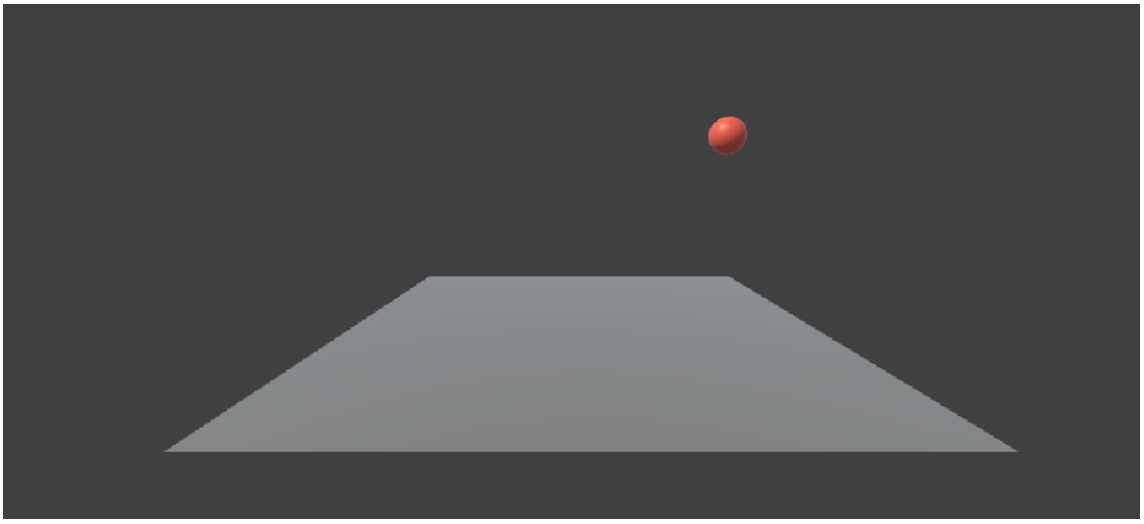
The implementation of this project focuses on integrating 3D modeling and Virtual Reality (VR) to enhance teaching, learning, and scientific simulation. By creating an immersive platform, the project aims to help students visualize complex concepts and perform virtual experiments. Through VR technology, users can interact with 3D models and simulate physical phenomena, offering a more engaging and effective way to understand challenging topics.

The project implementation will involve developing interactive 3D models and a user-friendly interface tailored to educational content in subjects like physics, chemistry, and biology. The VR environment will simulate real-world scientific scenarios, such as lab experiments and anatomical studies. The system will be optimized for both PC and VR hardware, ensuring accessibility, and real-time simulations will allow students to manipulate variables and observe outcomes. Testing throughout the process will ensure functionality, usability, and performance align with educational objectives.

6.1 Canon Model

The project implementation for the Canon model created in Unity focuses on delivering an interactive and immersive learning experience to teach basic physics concepts such as projectile motion and force dynamics. Using Unity's powerful engine, the model simulates the launch of a canon, allowing students to manipulate parameters like angle, velocity, and force to observe how these variables affect the trajectory of the canon ball in a virtual environment. This approach provides students with a hands-on learning experience, enabling them to visualize and interact with the fundamental principles of physics in a way that traditional textbooks cannot.

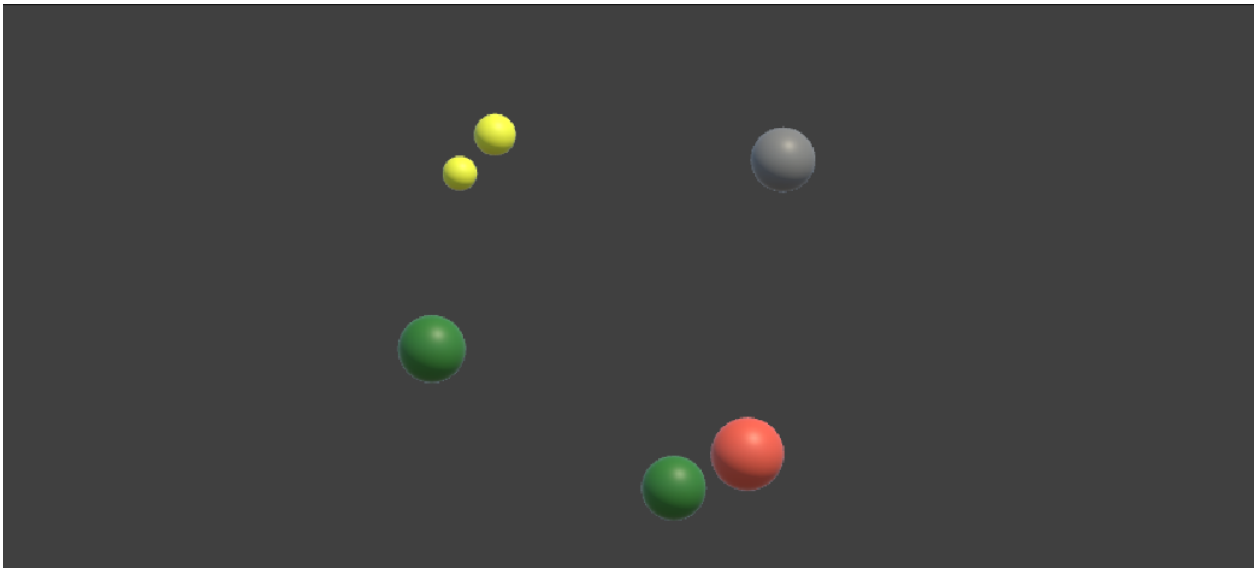
The implementation involves the creation of a 3D model of the canon and its environment within Unity, followed by programming the physics behind the projectile motion using Unity's built-in physics engine. The user interface will allow students to adjust parameters and launch the canon, while real-time visual feedback will show how changes impact the projectile's path, speed, and distance. The project also includes data visualization tools to display key information, such as speed, height, and distance, at various points in the simulation. Testing and optimization are key components of the process, ensuring the model runs smoothly across various platforms and provides accurate and educational simulations for users.



6.2 Input Module Implementation

The project implementation for the Constraint Dynamics model created in Unity aims to teach fundamental physics concepts related to forces, motion, and constraints in a virtual environment. Using Unity's physics engine, the model simulates the behavior of objects under constraints, such as ropes, pulleys, and springs, allowing students to manipulate these constraints and observe their effects on the system's dynamics. This interactive approach provides a deeper understanding of how forces are distributed, how motion occurs under certain limitations, and the real-world applications of these principles.

The implementation involves creating a 3D environment in Unity where various constraint systems are modeled, such as objects connected by ropes or pulleys. The physics engine will handle the interactions, applying real-time forces and constraints to the objects based on user inputs. Students can adjust parameters like the tension in ropes or the spring constant in a spring, observing how these changes affect the motion of the system. Visual feedback will display forces, motion paths, and object positions, enhancing understanding. The project will undergo thorough testing to ensure the accuracy of the simulations, smooth performance across different platforms, and an intuitive user experience for learners to explore and experiment with dynamic systems

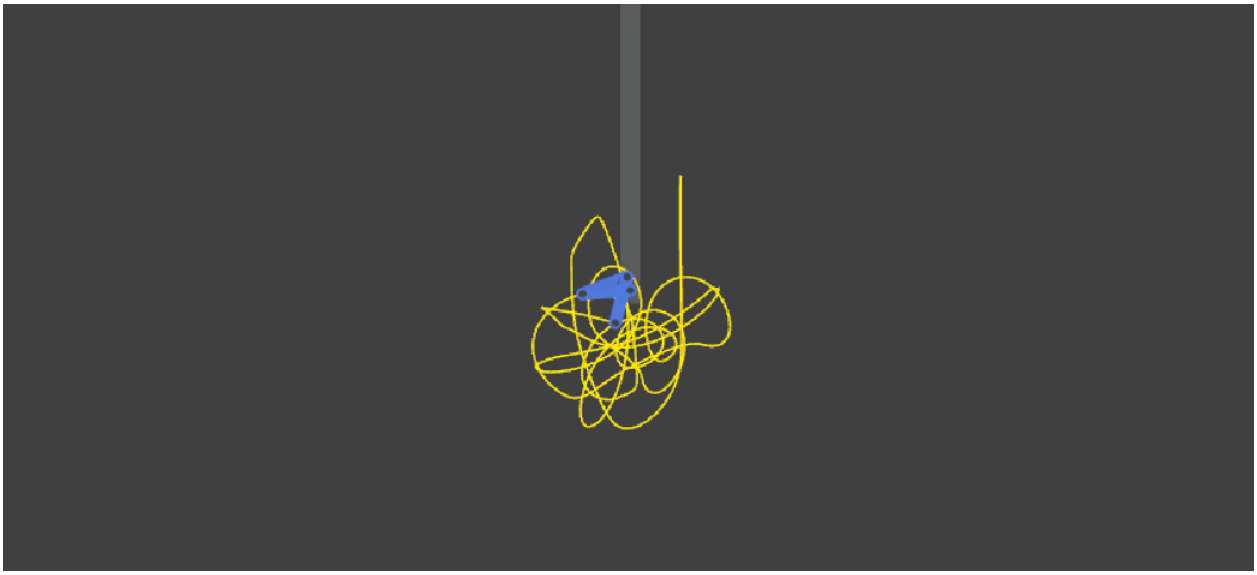


6.3 Triple Pendulum Model

The project implementation for the Triple Pendulum model created in Unity focuses on demonstrating the complex dynamics of a chaotic system in physics. A triple pendulum consists of three pendulums attached end-to-end, where the motion of each pendulum is influenced by the other two, creating intricate, nonlinear behavior. By simulating this system in Unity, students can interactively explore the chaotic motion, study concepts like angular momentum, energy conservation, and chaotic dynamics, and observe how small changes in initial conditions can lead to vastly different outcomes.

The implementation involves building a 3D model of the triple pendulum and using Unity's physics engine to simulate the motion of the three interconnected pendulums. The system will calculate the forces, torques, and angular displacements for each pendulum based on user-defined initial conditions, such as the starting angles and velocities. Real-time visual feedback will show

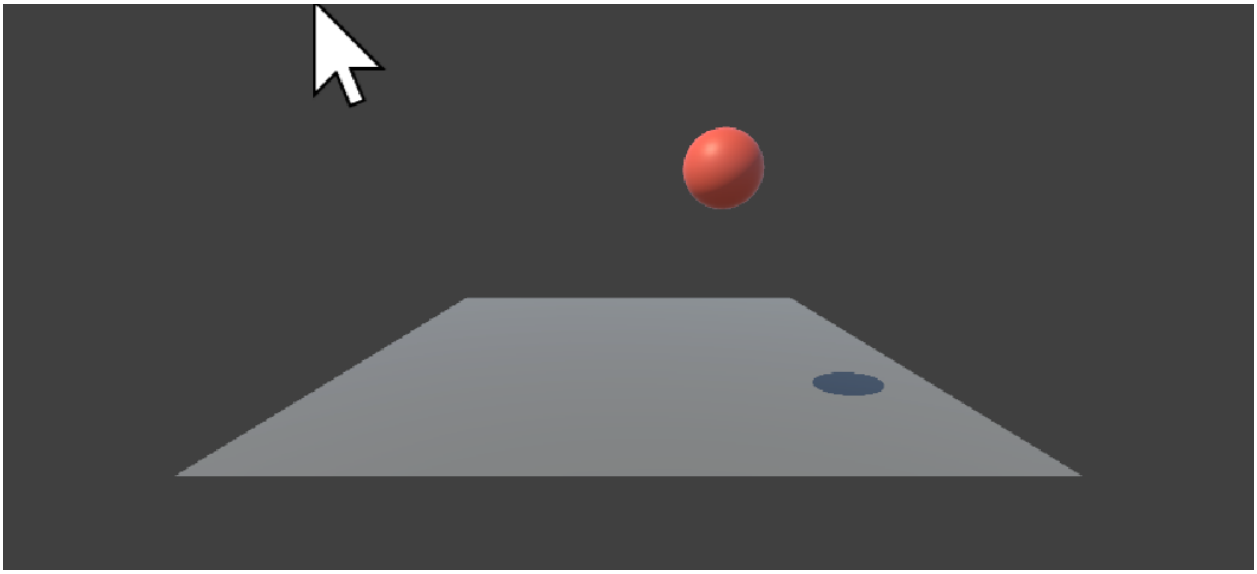
the motion paths and the interactions between the pendulums, helping students visualize the complexity and unpredictability of chaotic systems. Additionally, data visualization features will display relevant metrics such as energy conservation and angular velocities. The project will undergo rigorous testing to ensure that the simulation is accurate, the performance is optimized for different platforms, and the user interface is intuitive for students to explore and understand the principles of chaotic motion.



6.4 User Interactive Model

The project implementation for the User Interactive Ball model created in Unity focuses on providing an engaging and interactive experience for students to learn basic physics principles through direct manipulation of a ball. The model allows users to interact with a virtual ball by adjusting factors such as velocity, direction, force, and gravity, observing how these changes impact the ball's motion. This hands-on approach enables learners to visualize and experiment with concepts like projectile motion, collisions, and energy conservation in a dynamic and immersive 3D environment.

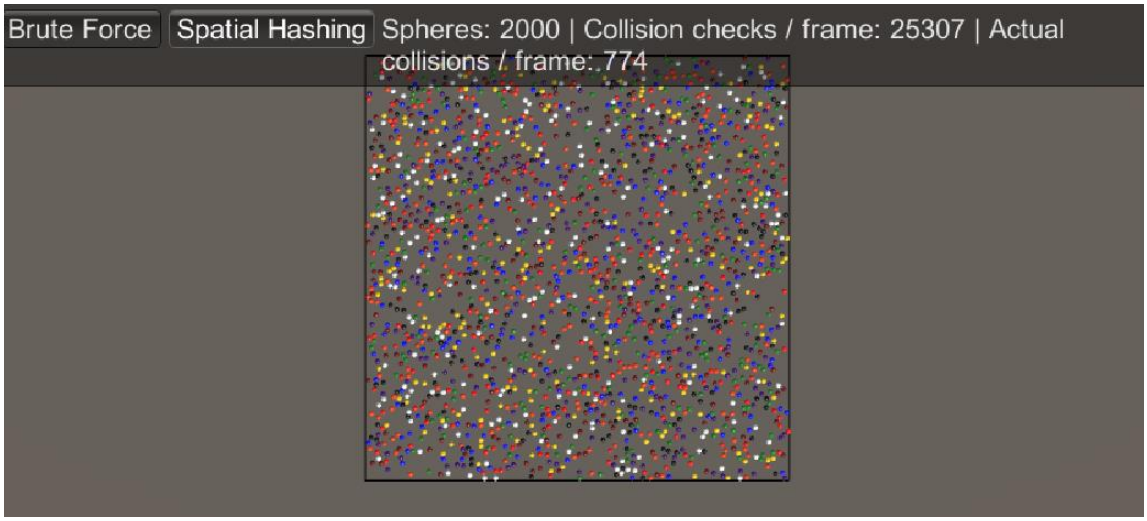
The implementation involves creating a 3D ball object within Unity, programming the physics using the Unity physics engine to simulate realistic movement and collisions. The user interface will allow users to modify key parameters such as the ball’s initial speed, launch angle, and environmental factors like gravity or wind. Through real-time interaction, users can observe the ball’s trajectory and behavior, gaining insights into how various forces influence its motion. The model will also include visual feedback to track the ball’s position, velocity, and acceleration at different points in time. Additionally, the project will feature interactive tutorials and challenges to guide users in applying physics concepts and exploring real-world applications. The project will undergo thorough testing to ensure smooth performance and accurate physics simulations across different platforms.



6.5 Fluid Physics

The project implementation for the custom Physics model of fluids, designed to represent materials like sand and snow, focuses on simulating granular and semi-solid fluid dynamics. This model, created in Unity, aims to teach students about fluid mechanics, particularly the behavior of non-Newtonian fluids or granular materials such as sand and snow, which display unique flow characteristics under different conditions. By interacting with the simulation, students can observe how these materials behave under gravity, pressure, and external forces, helping them understand key concepts like viscosity, porosity, and flow patterns in a virtual environment.

The implementation involves creating particle-based systems for sand and snow within Unity, where each particle behaves according to fluid and granular dynamics principles. Using Unity's physics engine and custom fluid simulation techniques, the particles representing sand or snow will interact with each other and the environment, simulating how these materials form piles, flow, and settle under different conditions. For example, sand particles will be modeled to move and pile up, while snow can simulate more cohesive properties, sticking together under certain conditions. The user will be able to interact with the model, applying forces, adjusting material properties like cohesion or friction, and observing how the behavior changes. Additionally, the model will provide visual feedback on the flow, particle dynamics, and interactions within the material, enhancing understanding of fluid and granular mechanics. The project will be thoroughly tested to ensure that the fluid behavior is realistic, the simulation runs efficiently across different devices, and the user interface allows easy interaction for educational purposes.



6.6 Eulerian Fluid Simulator

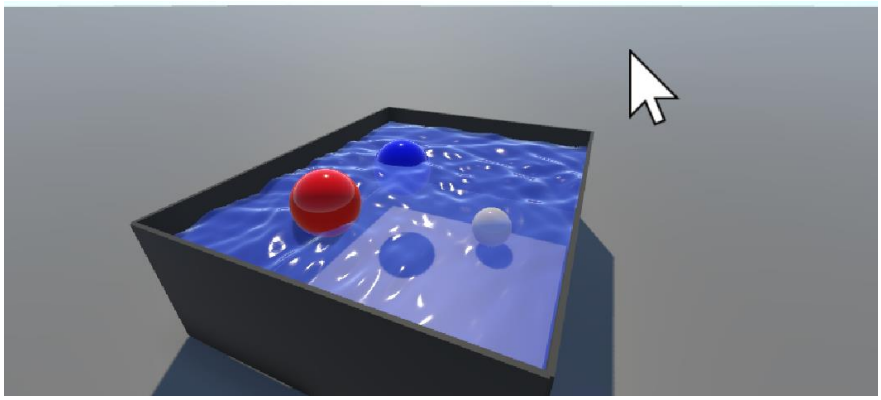
The project implementation of the Eulerian Fluid Simulator in Unity aims to simulate fluid dynamics based on the Eulerian approach, where the fluid is observed at fixed points in space, and the flow characteristics (such as velocity and pressure) are computed at those points over time. Unlike the Lagrangian approach, where individual fluid particles are tracked, the Eulerian method focuses on the behavior of fluid at different grid points or cells in a computational domain. This simulation allows students to visualize how fluids flow through different environments and how properties like pressure, velocity, and temperature change over time, offering a deeper understanding of fluid mechanics.

In Unity, the Eulerian Fluid Simulator is implemented by discretizing the fluid domain into a grid, with each cell representing a fixed location in the space where fluid characteristics are calculated. Using the Navier-Stokes equations (the fundamental equations of fluid dynamics) as the basis, the model simulates how fluid moves and interacts within the grid, including handling boundary conditions such as walls or obstacles. The user will be able to interact with the simulation by applying external forces like wind, pressure, or obstacles and observing how the fluid responds. For instance, they can see how fluid velocity changes when interacting with barriers or flowing through different sections. The visual representation will display fluid movement through color-coded gradients for velocity or pressure, and the fluid can be manipulated interactively to see real-time changes. This implementation provides a hands-on experience of fluid flow dynamics, helping students understand concepts like turbulence, laminar flow, and conservation of mass. The system will undergo rigorous testing to ensure accuracy in fluid behavior and responsiveness to user input, and it will be optimized for smooth performance across various platforms.

6.7 Buoyancy and Wave generator Model

The Buoyancy and Wave Generator Model implemented in Unity simulates the fundamental principles of buoyancy and wave dynamics in an interactive and visually intuitive environment. This model demonstrates how objects behave when placed in fluids and how waves are generated, propagate, and interact with floating bodies. The primary objective is to help users understand Archimedes' principle, displacement, buoyant forces, and wave motion concepts such as amplitude, frequency, and wavelength. This is particularly valuable in physics education, where visualizing these phenomena can significantly enhance conceptual clarity.

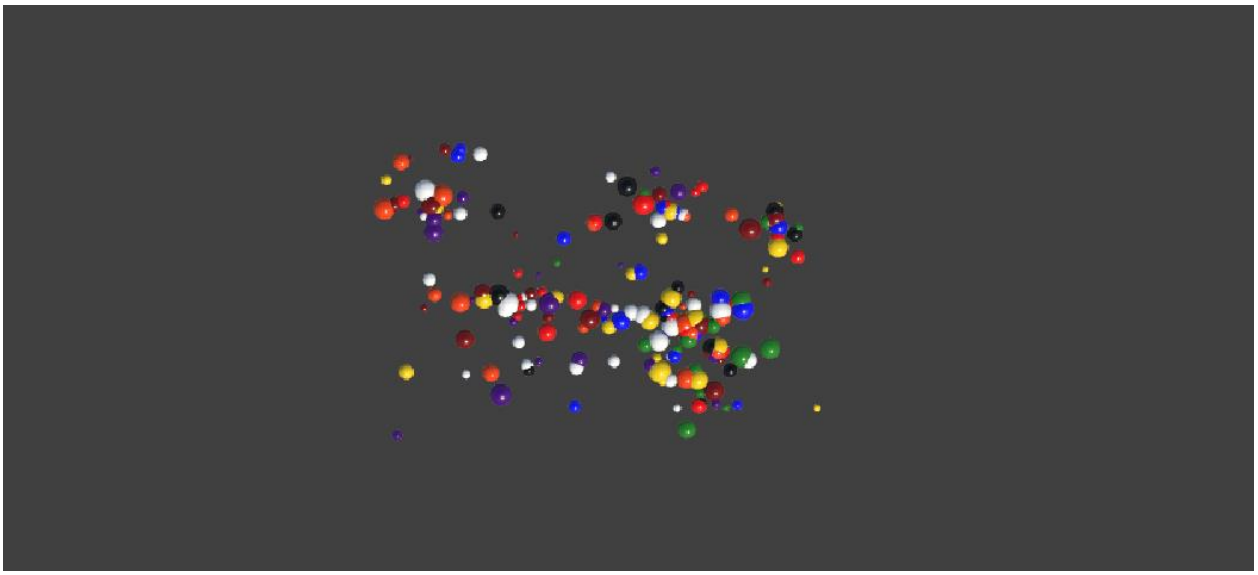
The implementation includes floating objects of various densities and shapes that respond realistically to wave interactions and buoyant forces. Unity's physics engine is used to simulate rigidbody dynamics and fluid surface interactions, while a procedural wave generator creates realistic wave patterns using sine functions or height maps. Users can adjust variables such as fluid density, object mass, and wave intensity to observe different behaviors. The model also allows for wave source customization, showing how disturbances affect the water surface and floating objects. This interactive approach makes it easier for students to experiment and learn through observation, reinforcing theoretical principles with hands-on experience. The model is designed to be scalable and responsive, providing consistent performance and clear visual feedback across platforms.



6.8 3-Body Problem

The 3-Body Problem Model developed in Unity is an interactive simulation that visually demonstrates the complex gravitational interactions between three celestial bodies. Unlike the two-body problem, which has precise analytical solutions, the three-body problem is known for its chaotic and unpredictable behavior. This model is designed to help learners explore the fundamental concepts of Newtonian mechanics, orbital motion, and chaos theory in an intuitive and engaging way.

The simulation uses Unity's physics system to model gravitational forces and motion in real-time. Users can place three bodies with configurable masses, initial velocities, and positions. Once the simulation begins, the system calculates gravitational attraction between each pair of bodies and updates their trajectories accordingly. Due to the chaotic nature of the system, even small changes in initial conditions can lead to drastically different outcomes—this property is effectively illustrated by the model. Users can pause, reset, or adjust parameters during runtime to observe multiple scenarios. The visual trajectories and real-time feedback offer a compelling way to grasp the complexities of gravitational dynamics beyond what static textbooks provide.

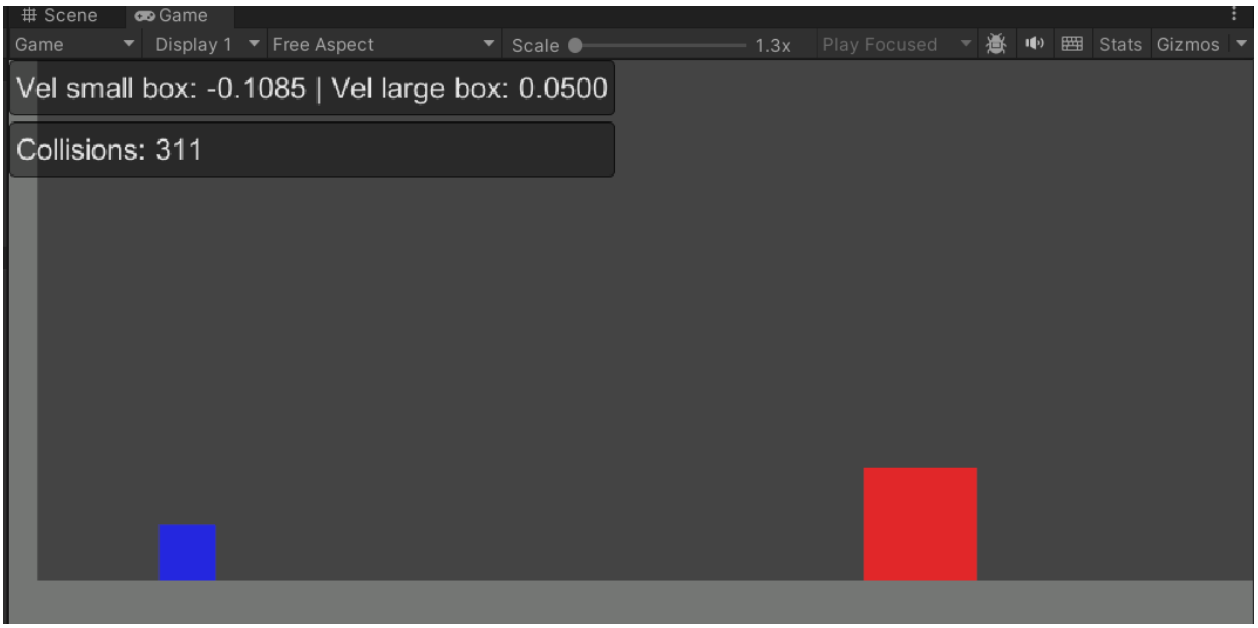


6.9 Pi- Problem

The Colliding Block Pi Problem Model developed in Unity is an elegant physics simulation that visualizes a surprising mathematical connection between elastic collisions and the digits of π (pi). Based on a classical thought experiment, the model features two blocks of different masses sliding on a frictionless surface and colliding elastically with each other and a wall. The number of collisions that occur before the system settles correlates with the digits of π, depending on the mass ratio of the blocks.

In this interactive Unity simulation, users can adjust the mass ratio between the two blocks—typically setting one block's mass to be a power of 100 times the other (e.g., 100, 10,000, etc.). As the blocks collide, both with each other and with a wall, the system meticulously counts the number of collisions. Remarkably, the total number of collisions aligns with the digits of π. For instance, a 100:1 mass ratio results in 31 collisions, which corresponds to "3.1", and increasing the ratio reveals more digits. This counterintuitive outcome is both visually engaging and intellectually

stimulating, offering a fun, hands-on way to explore the link between physics, math, and infinite numbers. The model effectively demonstrates conservation of momentum and energy while revealing the hidden beauty of mathematical patterns in physical systems.



7. RESULTS AND DISCUSSION

The result of the project is a comprehensive and interactive physics simulation platform built using Unity, designed to visually and intuitively teach fundamental and advanced physics concepts. Through models like the cannon, triple pendulum, user-interactive ball, Eulerian fluid simulator, and the colliding block π problem, the project successfully demonstrates complex theories such as motion, fluid dynamics, buoyancy, wave behavior, and chaotic systems. Each simulation enables students to explore physics in a hands-on virtual environment, bridging the gap between abstract theory and real-world behavior. The engaging visuals and interactivity not only enhance understanding but also spark curiosity and deeper interest in the subject.

7.1 Evaluation Parameters

The following key performance indicators (KPIs) were used to assess the system:

1. **Educational Effectiveness** – Measures how well the simulations aid in understanding and retaining physics concepts among students.
2. **User Engagement** – Evaluates how interactive and engaging the simulations are from the user's perspective.
3. **Accuracy of Simulation** – Assesses how closely the simulated models represent real-world physics behaviors and laws.
4. **Performance and Responsiveness** – Tests how smoothly the simulations run across various hardware setups, including loading times and frame rates.
5. **Usability and Interface Design** – Reviews the ease of navigation, clarity of controls, and overall user experience.
6. **Scalability and Extensibility** – Determines the ability to expand or enhance the platform with additional simulations or features.
7. **Platform Compatibility** – Checks the consistency of performance across different operating systems and devices (PC, tablet, etc.).
8. **Error Handling and Stability** – Observes how well the application manages bugs, crashes, or unexpected user inputs.
9. **Feedback and Improvement** – Gathers user and educator feedback to assess how the tool evolves based on iterative enhancements.
10. **Technical Completeness** – Evaluates whether all intended features and modules were implemented as planned.

7.2 Qualitative Analysis

The **real-world usability** of the system was tested by allowing **20 participants** to interact with it over multiple sessions.

7.2.1 Improved Conceptual Understanding

The project enables students to visualize and interact with abstract physics concepts, enhancing their understanding through experiential learning

7.2.2 High User Engagement

Interactive models and real-time feedback make the learning process more engaging, encouraging users to experiment and explore.

7.2.3 Versatility in Application

The project’s modular nature allows it to be used in classrooms, self-study environments, and even online courses with minimal modification.

7.2.4 Encouragement of Curiosity and Exploration

The open-ended nature of some models (like the user-interactive ball or fluid sandbox) encourages users to ask questions and explore outcomes on their own.

7.4 Technical Discussion

7.4.1 Development Platform and Tools

The entire suite of simulations has been developed using Unity Engine, chosen for its powerful physics engine, cross-platform capabilities, and real-time rendering performance. Unity's in-built Rigidbody and Collider systems, along with its flexibility in scripting with C#, allow for precise control over simulation parameters and interaction mechanics.

7.4.2 Physics Engine Utilization

The models extensively use Unity’s PhysX engine for simulating realistic physics behaviors—such as motion, gravity, collisions, and forces. Advanced models like the Eulerian Fluid Simulator and Buoyancy Wave Generator required integration with custom physics scripts to override or extend Unity’s native behavior, enabling more accurate fluid and wave representations

7.4.3 Custom Scripts and Algorithms

Key simulations like the Triple Pendulum, 3-Body Problem, and Unsolved Pi Collision Model were built with custom mathematical logic. These modules incorporate numerical integration techniques such as the Runge-Kutta method to manage non-linear, chaotic systems over time. Efficiency and accuracy in computation were a priority to maintain real-time simulation responsiveness.

7.4.4 Physics Engine Utilization

User interactions—like dragging, shooting, or modifying parameters—were built using Unity's Event System, raycasting, and custom UI elements. Interactive simulations (e.g., the ball movement and fluid sandbox) required consistent frame-rate handling and input sensitivity tuning to avoid lags and provide smooth experiences.

7.5 Limitations

Despite its many strengths, the system has a few limitations:

- **Hardware Dependency** – Complex simulations may lag or perform poorly on low-end devices.
- **Limited Real-World Accuracy** – Simplified physics models may not capture all real-world variables.
- **Scalability Constraints** – Adding highly detailed simulations can strain performance and memory.
- **User Learning Curve** – Some models require prior knowledge to understand interaction mechanics.
- **No Multi-User Support** – Currently designed for single-user interaction, lacking collaborative features.

7.7 Future Scope

The project can evolve significantly with enhancements in interactivity, accuracy, and accessibility. One promising direction is the integration of augmented and virtual reality (AR/VR), allowing students to immerse themselves in physics concepts and interact with models in three-dimensional space. This would create a more intuitive and engaging learning experience.

Additionally, the project can expand to support cloud-based deployment and multi-user collaboration, enabling students and educators to share, compare, and analyze simulations in real-time. Incorporating machine learning algorithms to adapt and personalize learning paths based on student performance is another exciting avenue. Lastly, future iterations can include a larger library of physics models, spanning topics from quantum mechanics to astrophysics, further enriching the educational potential of the platform.

8. CONCLUSION AND FUTURE ENHANCEMENTS

8.1 Conclusion

The development of the physics simulation project represents a significant stride towards transforming the way students perceive and engage with core physics concepts. By transitioning from traditional, often abstract, textbook methods to interactive and visual simulations, the project addresses the long-standing issue of conceptual clarity among learners. Through visually dynamic models, students gain a more intuitive understanding of physical phenomena, enhancing both interest and retention.

Throughout the implementation of this project, various models like the cannon simulator, triple pendulum, buoyancy generator, and Eulerian fluid simulator were meticulously designed and developed using Unity. These simulations bridge the gap between theoretical learning and practical visualization, enabling learners to observe real-time physical behaviors under different conditions. Each model serves a specific educational purpose, from demonstrating classical mechanics to fluid dynamics, ensuring comprehensive coverage of physics domains.

A user-centered approach played a crucial role in shaping the interactive design. The project has included responsive feedback mechanisms and intuitive controls, allowing users to manipulate parameters and observe outcomes. This interactivity not only deepens engagement but also fosters critical thinking and exploratory learning, crucial for scientific education.

The implementation process also reflected rigorous software development practices. Through systematic stages of planning, design, development, integration, and testing, the team ensured functionality, reliability, and scalability. The use of Unity as the development platform allowed for sophisticated simulations with high-quality rendering and physics engines, supporting both educational depth and technical stability.

Testing strategies like unit testing, integration testing, and user acceptance testing ensured the models met performance and usability standards. While minor constraints such as hardware requirements and simulation limits were encountered, the overall system proved robust and accessible across standard client devices. Feedback from trial users, especially students, indicated improved conceptual understanding and increased motivation to learn physics.

Despite certain limitations—like the scope of covered topics and hardware performance bottlenecks—the project has established a strong foundation for future expansion. The system's modular architecture and open-ended design allow for seamless integration of new models, features, and technologies like AR/VR. Moreover, educational institutions can adopt and adapt the platform to suit varying curricula and pedagogical needs.

In conclusion, this project not only serves as a powerful learning tool but also showcases the potential of technology in reshaping science education. By making physics more accessible, engaging, and interactive, it fosters a new era of experiential learning. Continued research, iterative improvement, and broader implementation can turn this platform into a cornerstone of modern STEM education.

8.2 Future Enhancements

While the project has shown promising results and practical usability, there are several areas in which it can be expanded, enhanced, or optimized further. These enhancements could significantly improve the system's reliability, flexibility, and application scope in real-world environments.

1. Augmented Reality (AR) Integration:

Integrating AR capabilities would enable students to visualize simulations in their physical environment through mobile or AR glasses, creating a more immersive and interactive learning experience. This would help students relate physical concepts to the real world.

2. Virtual Reality (VR) Support:

Incorporating VR technology can take the interactive simulations to the next level by allowing users to step into a completely immersive 3D environment. Students could interact with models in a more lifelike manner, offering a deeper understanding of concepts like gravity, forces, or fluid dynamics.

3. Adaptive Learning Algorithms:

Implementing AI-based adaptive learning systems that tailor simulations and challenges based on the user's performance would personalize the experience. It would ensure students progress through topics at their own pace and receive instant feedback on areas where they struggle.

4. Real-Time Collaboration Features:

Adding collaborative functionalities would allow multiple users to interact with the simulations simultaneously, facilitating group work and real-time problem-solving. Students could work together from different locations, promoting remote learning and teamwork.

5. Expanded Model Library:

A larger variety of physics models could be developed and added, covering more advanced topics like thermodynamics, electromagnetism, or quantum mechanics. This would provide a broader range of educational tools for high school and college-level students.

6. Multi-Device Support:

Expanding compatibility to allow the simulations to run seamlessly across multiple devices (e.g., smartphones, tablets, and desktops) would make the project more accessible to a wider audience. Cloud-based solutions could ensure progress is synced across devices.

7. Enhanced Physics Engine:

Upgrading the physics engine to simulate more complex phenomena with higher accuracy—such as fluid turbulence, electromagnetic fields, or realistic material properties—would improve the realism of the models, offering a more sophisticated learning tool.

8. Gamification Elements:

Introducing gamified elements, such as scoring, levels, and achievements, could further increase engagement. This would transform the learning process into a more enjoyable and motivating experience, appealing to a broader demographic, including younger students.

9. Mobile App Development:

Developing a dedicated mobile app for both Android and iOS platforms would allow users to access simulations anytime and anywhere. Mobile accessibility would increase the reach of the project, particularly in areas where traditional computer resources may be limited.

10. Support for Multiple Languages:

Implementing multilingual support would make the system accessible to a global audience. Offering simulations in various languages would break down language barriers, enhancing the educational value for students worldwide.

8.3 Broader Applications

The broader application of the physics simulation project extends beyond just classroom learning. It has the potential to be utilized in various fields, such as engineering education, scientific research, and even entertainment. For engineering students, simulations like fluid dynamics or mechanics can provide a deeper understanding of complex systems, helping bridge the gap between theory and real-world applications. In scientific research, such simulations can serve as powerful tools for visualizing and testing hypotheses about physical phenomena that might be difficult to observe or replicate in a lab environment. Furthermore, the entertainment industry, especially in game development and animation, can leverage the project's physics models to create more realistic and engaging virtual environments. By expanding its application to various sectors, the project can play a significant role in advancing education, research, and technology across multiple disciplines.

8.4 Final Thoughts

In final thoughts, this project demonstrates the transformative potential of integrating virtual reality and 3D simulations in educational contexts, particularly in understanding complex physics concepts. The interactive nature of the models allows for a more hands-on and immersive learning experience, bridging the gap between theoretical knowledge and real-world application. The project not only enhances the understanding of physical phenomena but also fosters critical thinking, problem-solving, and curiosity in students, preparing them for future scientific and technological challenges.

Looking ahead, the potential for further enhancements and applications is vast, especially as technology evolves. Integration with AI, real-time physics engines, and more advanced VR systems could further refine the realism and interactivity of simulations, pushing the boundaries of how we teach and study physics. The future scope of the project could lead to widespread adoption in educational institutions globally, fostering a more engaged, interactive, and inclusive learning environment for students across various disciplines.

9. REFERENCES

1. Stern, D., & Johnson, E. (2018). Virtual Reality for Education: From Passive Learning to Immersive Interaction. Journal of Educational Technology, 23(4), 512-526.

2. Azuma, R. T. (1997). A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments, 6(4), 355-385.

3. Slater, M., & Wilbur, S. (1997). A Framework for Immersive Virtual Environments (FIVE): Speculations on the Role of Presence in Virtual Environments. ACM SIGGRAPH, 3(2), 29-35.

4. Friedman, J., & Hagler, M. (2007). 3D Modeling and Simulation in Scientific Research. Journal of Computational Physics, 23(1), 1-11.

5. Macedonia, M. (2002). Immersive Learning Environments: A Simulation-Driven Approach. International Journal of Engineering Education, 18(2), 137-146.

6. Mitchell, J., & Beck, H. (2016). Integration of 3D Modeling and Virtual Reality into the Science Curriculum. Journal of Science Education and Technology, 25(6), 827-841.

7. Hendrix, C., & Benassi, V. (2004). Computer-Based Physics Simulations: An Innovative Approach to Science Education. Physics Education, 39(2), 107-112.

8. Dede, C. (2009). Immersive Interfaces for Engagement and Learning. Science, 323(5910), 66-69.

9. Bailenson, J. N., & Beall, A. C. (2006). Transformed Social Interaction in Collaborative Virtual Environments. Proceedings of the International Conference on Human-Computer Interaction, 157-162.

10. Gunter, L., & Greenberg, D. (2008). The Role of Virtual Reality in Teaching Science. Journal of Science Education, 29(4), 315-323.

11. Schmidt, M., & Epple, M. (2019). Virtual Reality: An Innovative Approach for Learning Physics in High School. Physics Teacher, 57(3), 184-188.

12. Parisi, D., & Thomas, L. (2007). 3D Visualization in Scientific Research and Education. Computers & Education, 48(2), 124-135.

13. Plomp, T., & Nieveen, N. (2010). Educational Design Research: An Introduction. International Journal of Design in Learning, 6(1), 1-16.

14. Barker, T., & Hayward, B. (2010). Using 3D Interactive Models in Science Education. Journal of Educational Multimedia and Hypermedia, 19(2), 157-167.

15. Anderson, C., & Shih, T. (2017). Teaching Physics with Virtual Reality: A New Frontier for High School Education. Physics Education Research Conference Proceedings, 162-167.

16. Wright, D., & Gunning, D. (2018). Leveraging Virtual Reality for Teaching Physics in the Classroom. Innovative Teaching Technology Journal, 35(2), 55-62.

17. Wu, H., & Lee, J. (2014). The Impact of Virtual Reality on Learning Physics: A Pilot Study. Education and Information Technologies, 19(2), 319-334.

18. Kennedy, R., & Jones, A. (2015). Virtual Reality and Augmented Reality for Education: A Review. Learning Technologies Research Journal, 24(3), 41-56.

19. Iglehart, R., & Davis, D. (2002). Use of Simulation and Modeling to Teach Physics. Physics Education Research, 47(3), 110-118.

20. Hegarty, M., & Sims, V. (2012). Learning from 3D Models: The Role of Spatial Skills and Visualization. Journal of Educational Psychology, 104(2), 387-398.

APPENDIX A – SAMPLE CODE

Cannon Controller file:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CannonController : MonoBehaviour
{


    public Transform ballTrans;

    private Vector3 ballVel;
    private Vector3 ballPos;

    private float ballRadius;

    private int subSteps = 5;

    private Vector3 gravity = new Vector3(0f, -9.81f, 0f);



    private void Start()
    {
        //Init pos and vel
        ballPos = ballTrans.position;

        ballVel = new Vector3(3f, 5f, 2f);

        ballRadius = ballTrans.localScale.y * 0.5f;
    }



    private void Update()
    {

        ballTrans.position = ballPos;
    }
```

```
private void FixedUpdate()
{
    float sdt = Time.fixedDeltaTime / (float)subSteps;

    for (int i = 0; i < subSteps; i++)
    {
        ballVel += gravity * sdt;
        ballPos += ballVel * sdt;
    }



    float halfSimSize = 5f - ballRadius;

    //x
    if (ballPos.x < -halfSimSize)
    {
        ballPos.x = -halfSimSize;
        ballVel.x *= -1f;
    }
    else if (ballPos.x > halfSimSize)
    {
        ballPos.x = halfSimSize;
        ballVel.x *= -1f;
    }

    //y
    if (ballPos.y < 0f + ballRadius)
    {
        ballPos.y = 0f + ballRadius;
        ballVel.y *= -1f;
    }


    //z
    if (ballPos.z < -halfSimSize)
    {
        ballPos.z = -halfSimSize;
        ballVel.z *= -1f;
    }
    else if (ballPos.z > halfSimSize)
    {
        ballPos.z = halfSimSize;
        ballVel.z *= -1f;
    }
```

```
    }
}
```

Stimulate Constraint:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Constraints;

public class ConstraintsController : MonoBehaviour
{

    public GameObject beadGO;

    public List<Material> materials;

    private Vector3 wireCenter = Vector3.zero;

    private float wireRadius = 5f;

    private Vector3 gravity = new Vector3(0f, -9.81f, 0f);

    private float restitution = 1f;

    private int subSteps = 20;

    private List<Bead> allBeads;



    private void Start()
    {
        //this.bead = new Bead(1f, ballGO.transform.position);

        ResetSimulation();
    }


    private void ResetSimulation()
    {
        allBeads = new List<Bead>();

        for (int i = 0; i < 6; i++)
```

```
    {
        GameObject newBallGO = Instantiate(beadGO);


        Vector2 posOnCircle = Random.insideUnitCircle.normalized * 5f;

        Vector3 randomPos = new Vector3(posOnCircle.x, posOnCircle.y, 0f);

        float randomSize = Random.Range(0.5f, 2f);

        newBallGO.transform.position = randomPos;
        newBallGO.transform.localScale = Vector3.one * randomSize;

        Bead newBead = new Bead(newBallGO.transform);

        //Random material
        Material randomMat = materials[Random.Range(0, materials.Count)];

        newBallGO.GetComponent<MeshRenderer>().sharedMaterial = randomMat;

        allBeads.Add(newBead);
    }
}



private void Update()
{
    foreach (Bead b in allBeads)
    {
        b.UpdateVisualPosition();
    }
}



private void FixedUpdate()
{
    float dt = Time.fixedDeltaTime;

    float sdt = dt / (float)subSteps;

    for (int step = 0; step < subSteps; step++)
    {
        for (int i = 0; i < allBeads.Count; i++)
        {
```

```
            allBeads[i].StartStep(sdt, gravity);
        }

        for (int i = 0; i < allBeads.Count; i++)
        {
            allBeads[i].KeepOnWire(wireCenter, wireRadius);
        }

        for (int i = 0; i < allBeads.Count; i++)
        {
            allBeads[i].EndStep(sdt);
        }

        for (int i = 0; i < allBeads.Count; i++)
        {
            for (int j = i + 1; j < allBeads.Count; j++)
            {
                BallCollisionHandling.HandleBallBallCollision(allBeads[i], allBeads[j],
restitution);
            }
        }
    }
  }



  private void LateUpdate()
  {

    DisplayShapes.DrawCircle(wireCenter, wireRadius, DisplayShapes.ColorOptions.White,
DisplayShapes.Space2D.XY);
  }
}
```

N- Pendulum code:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class NPendulumSimulatorDouble
{

  public readonly List<NodeDouble> pendulumSections = new List<NodeDouble>();
```

```
    private readonly double pendulumLength;

    private readonly int numberOfPendulumSections;

    private double SectionLength => pendulumLength / (double)numberOfPendulumSections;

    private readonly Vector3Double gravity = new (0.0, -9.81, 0.0);

    private readonly int seed = 0;


    public NPendulumSimulatorDouble(int numberOfPendulumSections, double length,
Vector3 startPos, float startAngleOffset = 0)
    {
        this.numberOfPendulumSections = numberOfPendulumSections;

        this.pendulumLength = length;


        Random.InitState(seed);


        Vector3Double startPosDouble = new (startPos.x, startPos.y, startPos.z);

        NodeDouble wallSection = new (startPosDouble, 0.0, true);

        pendulumSections.Add(wallSection);

        Vector3 pendulumStartDir = new Vector3(1.0f, 0.6f, 0.0f).normalized;

        for (int n = 0; n < numberOfPendulumSections; n++)
        {
            startPos += pendulumStartDir * (float)SectionLength;

            float mass = 0.5f;


            startPosDouble = new Vector3Double(startPos.x, startPos.y, startPos.z);

            NodeDouble newSection = new NodeDouble(startPosDouble, mass, false);

            pendulumSections.Add(newSection);
```

```
        float randomAngleZ = Random.Range(0f, 85f);

        randomAngleZ += startAngleOffset;

        pendulumStartDir = Quaternion.Euler(0f, 0f, randomAngleZ) * pendulumStartDir;
    }
}



public void Simulate(double dt)
{



    for (int i = 1; i < pendulumSections.Count; i++)
    {
        NodeDouble thisNode = pendulumSections[i];

        thisNode.StartStep(dt, gravity);
    }


    for (int i = 1; i < pendulumSections.Count; i++)
    {
        NodeDouble prevNode = pendulumSections[i - 1]; //x1
        NodeDouble thisNode = pendulumSections[i]; //x2

        Vector3Double dir = thisNode.pos - prevNode.pos;

        double currentLength = dir.Magnitude;

        double w1 = !prevNode.isFixed ? 1.0 / prevNode.mass : 0.0;
        double w2 = !thisNode.isFixed ? 1.0 / thisNode.mass : 0.0;



        prevNode.pos -= w1 * correction * dir;
        thisNode.pos += w2 * correction * dir;
    }



    for (int i = 1; i < pendulumSections.Count; i++)
    {
```

```
            NodeDouble thisNode = pendulumSections[i];

            thisNode.EndStep(dt);
        }
    }
}
```

User Interactive Code:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UserInteraction;

public class UserInteractionController : MonoBehaviour
{
    public Transform ballTransform;

    public Texture2D cursorTexture;

    private InteractiveBall ball;

    private int subSteps = 5;

    private Vector3 gravity = new Vector3(0f, -9.81f, 0f);

    private Grabber grabber;



    private void Start()
    {
        //Init the ball
        ball = new InteractiveBall(ballTransform);

        ball.vel = new Vector3(3f, 5f, 2f);


        grabber = new Grabber(Camera.main);

        Cursor.visible = true;

        Cursor.SetCursor(cursorTexture, Vector2.zero, CursorMode.ForceSoftware);
```

```
        //Cursor.SetCursor(UnityEditor.PlayerSettings.defaultCursor,                Vector2.zero,
CursorMode.ForceSoftware);
    }



    private void Update()
    {
        ball.UpdateVisualPosition();

        grabber.MoveGrab();
    }



    private void LateUpdate()
    {
        if (Input.GetMouseButtonDown(0))
        {
            List<IGrabbable> temp = new List<IGrabbable>();

            temp.Add(ball);

            grabber.StartGrab(temp);
        }

        if (Input.GetMouseButtonUp(0))
        {
            grabber.EndGrab();
        }
    }



    private void FixedUpdate()
    {
        float sdt = Time.fixedDeltaTime / (float)subSteps;

        for (int step = 0; step < subSteps; step++)
        {
            ball.SimulateBall(sdt, gravity);
        }

        ball.HandleWallCollision();
    }
}
```

3 Body Problem:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


//The 3-body-problems with planets interacting with each other
//https://en.wikipedia.org/wiki/Three-body_problem
//You can add how many bodies you want, so its more like N-Body Problem
public class ThreeBodyProblemController : MonoBehaviour
{
    //
    // Public
    //

    public GameObject planetPrefabGO;


    //
    // Private
    //

    private const int SEED = 0;

    private Camera thisCamera;

    private readonly List<Planet> allPlanets = new();

    private readonly bool displayHistory = false;

    private readonly List<List<Vector3>> historicalPositions = new ();

    private readonly List<DisplayShapes.ColorOptions> historicalPositionsColor = new ();


    private const int NUMBER_OF_PLANETS = 200;

    private readonly MinMax minMaxPlanetRadius = new (0.1f, 0.4f);


    //Simulation settings
    private const int SUB_STEPS = 1;

    private readonly Vector2 mapSize = new(14f, 10f);
```

```
    private readonly MinMax minMaxRSqr = new(0.3f, 25f);



    private void Start()
    {
        Random.InitState(SEED);

        AddPlanets();

        GivePlanetsRandomColor();

        /*
        if (displayHistory && NUMBER_OF_PLANETS == 3)
        {
            //Assume we have 3 planets which is the 3-body-problem
            historicalPositions.Add(new List<Vector3>());
            historicalPositions.Add(new List<Vector3>());
            historicalPositions.Add(new List<Vector3>());

            historicalPositionsColor.Add(DisplayShapes.ColorOptions.Red);
            historicalPositionsColor.Add(DisplayShapes.ColorOptions.Blue);
            historicalPositionsColor.Add(DisplayShapes.ColorOptions.Yellow);

            allPlanets[0].ballTransform.GetComponent<MeshRenderer>().material.color          =
Color.red;
            allPlanets[1].ballTransform.GetComponent<MeshRenderer>().material.color          =
Color.blue;
            allPlanets[2].ballTransform.GetComponent<MeshRenderer>().material.color          =
Color.yellow;
        }
        */



        //}


        thisCamera = Camera.main;

        CenterCamera();
    }



    private void Update()
```

```
{
    foreach (Planet p in allPlanets)
    {
        p.UpdateVisualPosition();
    }

    ZoomCamera();

    if (displayHistory && NUMBER_OF_PLANETS == 3)
    {
        for (int i = 0; i < allPlanets.Count; i++)
        {
            historicalPositions[i].Add(allPlanets[i].pos);
        }
    }
}



private void LateUpdate()
{
    /*
    Vector3 center = GetCenterOfMass();

    List<Vector3> lineSegments = new();

    foreach (Planet p in allPlanets)
    {
        lineSegments.Add(p.pos);
        lineSegments.Add(center);
    }

    DisplayShapes.DrawLineSegments(lineSegments, DisplayShapes.ColorOptions.White);
    */

    if (displayHistory && NUMBER_OF_PLANETS == 3)
    {
        for (int i = 0; i < historicalPositions.Count; i++)
        {
            DisplayShapes.DrawLine(historicalPositions[i], historicalPositionsColor[i]);
        }
    }
}
```

```
private Vector3 GetCenterOfMass()
{
    Vector3 centerOfMass = Vector3.zero;

    float totalMass = 0f;

    foreach (Planet p in allPlanets)
    {
        centerOfMass += p.pos * p.mass;

        totalMass += p.mass;
    }

    centerOfMass /= totalMass;

    return centerOfMass;
}



private void CenterCamera()
{
    Vector3 center = GetCenterOfMass();

    Vector3 cameraPos = center;

    cameraPos.y = thisCamera.transform.position.y;

    thisCamera.transform.position = cameraPos;
}



private void ZoomCamera()
{
    bool isVisible = true;

    foreach (Planet p in allPlanets)
    {
        if (!p.ballTransform.GetComponent<Renderer>().isVisible)
        {
            isVisible = false;

            break;
        }
    }
```

```
    //if (!isVisible)
    //{
    //    Debug.Log("Zoom out");
    //}

    if (isVisible)
    {
        return;
    }

    float zoomSpeed = 0.5f;

    if (isVisible)
    {
        zoomSpeed *= -1f;
    }

    float size = thisCamera.orthographicSize;

    size += zoomSpeed * Time.deltaTime;

    size = Mathf.Clamp(size, 5f, 100f);

    Camera.main.orthographicSize = size;
}



private void FixedUpdate()
{
    Vector3[] accelerations = CalculateAccelerations();


    float sdt = Time.fixedDeltaTime / (float)SUB_STEPS;

    for (int i = 0; i < allPlanets.Count; i++)
    {
        Planet thisPlanet = allPlanets[i];

        thisPlanet.SimulatePlanet(SUB_STEPS, sdt, accelerations[i]);


    }
}
```

```
private Vector3[] CalculateAccelerations()
{
    Vector3[] accelerations = new Vector3[allPlanets.Count];

    for (int i = 0; i < allPlanets.Count; i++)
    {
        Planet thisPlanet = allPlanets[i];

        for (int j = i + 1; j < allPlanets.Count; j++)
        {
            Planet otherPlanet = allPlanets[j];


            float m1 = thisPlanet.mass;
            float m2 = otherPlanet.mass;

            Vector3 thisOtherVec = otherPlanet.pos - thisPlanet.pos;

            float rSqr = thisOtherVec.sqrMagnitude;

            rSqr = Mathf.Clamp(rSqr, minMaxRSqr.min, minMaxRSqr.max);

            float F = G * ((m1 * m2) / rSqr);

            //F = m * a
            float aThisPlanet = F / m1;
            float aOtherPlanet = F / m2;

            accelerations[i] += aThisPlanet * thisOtherVec.normalized;
            accelerations[j] += aOtherPlanet * -thisOtherVec.normalized;
        }
    }

    return accelerations;
}



private void AddPlanets()
{
    for (int i = 0; i < NUMBER_OF_PLANETS; i++)
    {
        GameObject newBallGO = GameObject.Instantiate(planetPrefabGO);
```

```
        float randomSize = Random.Range(minMaxPlanetRadius.min, minMaxPlanetRadius.max);

        newBallGO.transform.localScale = Vector3.one * randomSize;

        float halfBallSize = randomSize * 0.5f;

        float halfWidthX = mapSize.x * 0.5f;
        float halfWidthY = mapSize.y * 0.5f;

        float randomPosX = Random.Range(-halfWidthX + halfBallSize, halfWidthX - halfBallSize);
        float randomPosZ = Random.Range(-halfWidthY + halfBallSize, halfWidthY - halfBallSize);

        Vector3 randomPos = new(randomPosX, 0f, randomPosZ);

        newBallGO.transform.position = randomPos;

        Planet newPlanet = new(newBallGO.transform);

        allPlanets.Add(newPlanet);
      }
    }



    private void GivePlanetsRandomColor()
    {

      Material ballBaseMaterial = planetPrefabGO.GetComponent<MeshRenderer>().sharedMaterial;

      foreach (Planet p in allPlanets)
      {
        Material randomBallMaterial = Billiard.BilliardMaterials.GetRandomBilliardBallMaterial(ballBaseMaterial);

        p.ballTransform.GetComponent<MeshRenderer>().material = randomBallMaterial;
      }
    }
}
```